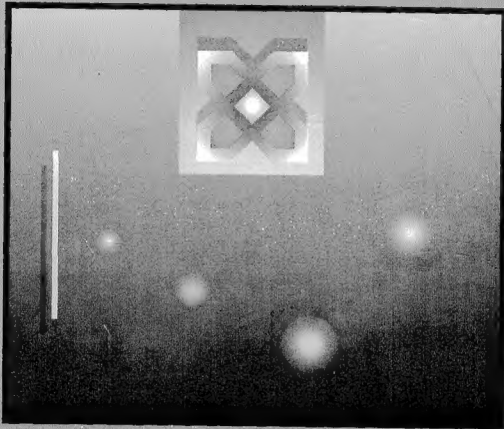


تطبيقات نظم إدارة قواعد البيانات



أ.د. محمد فهمي طلبة

أ.د. عمرو جنيّد

أ.د. محمد علي الشرقاوي

م.د. مصطفى رضا عبد الوهاب

أ.د. الدين محمد فهمي

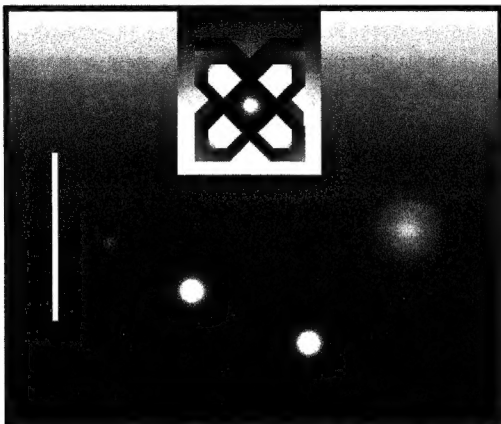
أ.د. فتيحة محمد إسماعيل



موسوعة دلتا كمبيوتر

تطبيقات نظم إدارة قواعد البيانات

تطبيقات نظم إدارة قواعد البيانات



١.د. محمد فهمى طلبه

١.د. عمرو جنيـد

١.د. محمد على الشرقاوى

م. مصطفى رضا عبد الوهاب

د. علاء الدين محمد فهمى

مصطفى محمد إسماعيل

٧

موسوعة دلتا كمبيوتر 

© حقوق النشر

لا يجوز نشر أى جزء من هذا الكتاب أو اختزان مادته بطريقة الاسترجاع ، أو نقله على أى وجه ، أو بأى طريقة ، سواء كانت إلكترونية ، أو ميكانيكية ، أو بالتصوير ، أو بالتسجيل ، أو خلاف ذلك إلا بموافقة الناشر على هذا كتابة ومقدمًا .

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior permission of the publisher.

رقم الإيداع : ٩١/٢٨٨٥

تقديم

ان السؤال الدائر بين المتخصصين فى مجال الحاسبات عن كون تخطيط البرامج على وجه الخصوص وبعض جوانب علوم الحاسب عامة تقع فى اطار العلم (Science) أو فى اطار الفن (Art) . وبدون الخوض فى تفاصيل تعريف كل منهما من ناحية الأسس القائم عليها وعلاقة ذلك بالانسان فانه يوجد شبه اتفاق بين كل الخبراء فى أن تخطيط البرامج وإن كان يركز على العديد من الأسس العلمية التى يجب الاتمام بها الا أنه يحتاج الى العديد من الجوانب المرتبطة بالفن والتى تتأثر بخبرة مخطط البرامج وقدرته الابداعية بالإضافة الى شخصيته .

ومما لاشك فيه أن البرنامج الواحد - الذى يؤدي مهمة معينة للوصول الى هدف أو أهداف محددة - اذا ما تم تنفيذه بواسطة العديد من مخططي البرامج فانه يصعب أن تتفق خطوات التنفيذ لأى منهم مع الآخر رغم أن النظام النهائى يؤدي نفس المهمة . ولا يعتبر الاختلاف فى النظم المنتجة ناشئا فقط من التباين فى الخلفية العلمية لمخططي البرامج بقدر ما يعبر عن التباين فى خبرتهم وشخصيتهم بالإضافة الى قدرتهم الابداعية .

ولا خلاف فى أن الخبرة والقدرة الابداعية يمكن أن تصقل بواسطة الممارسة بعد التقليد . ومن هذا البعد كانت فكرة هذا الكتاب الذى يعتبر من أحد أهدافه تمكين مخطط البرامج - الملم بكل الجوانب العلمية لهندسة تخطيط البرامج والمتمكن من أدوات البرمجة المستخدمة - من متابعة بعض النماذج للحلول المقترحة لمشاكل معينة بحيث تمكنه بصورة أو بأخرى من تقليدها فى أحد الاطارات الاخرى المشابهة . ومع التباين بين النظم التى تم مناقشتها يستطيع القارئ اكتساب بعض الخبرات فى العديد من المجالات المطروحة فى هذا الكتاب .

ويمكن للقارئ أن ينتقل من مرحلة الاستيعاب الى مرحلة التقليد ثم الى مرحلة التعديل يليها مرحلة التطوير ثم ينطلق الى مرحلة الابداع . وقد اختار هذا الكتاب

أحد أدوات البرمجة المتميزة وهي نظم برامج عائلة DBASE . وتنتمي الى هذه العائلة نظم برامج FOXPRO , FOXBASE , DBASE IV , DBASE III+ , ولا بد للقارئ عند الاطلاع على هذا الكتاب أن يكون ملماً بأوامر وأساليب البرمجة المستخدمة في لغة هذه العائلة . وجدير بالذكر أن كتاب نظم إدارة قواعد البيانات "الجزء الأول" و "الجزء الثاني" من موسوعة "دلتا كمبيوتر" لتكنولوجيا وعلوم الحاسب تمكن القارئ المبتدئ من التعرف على أوامر وأساليب البرمجة باستخدام أى من نظم برامج عائلة DBASE .

وفي مرحلة الاستيعاب يمكن للقارئ أن يركز في مضمون النظم الموضحة في أجزاء وفصول الكتاب المختلفة ، وأن يحدد وظيفة كل جزء من هذه الأجزاء ومن ثم التعرف على بعض الأفكار المطروحة والاطار العام لتصميم النظام وهيكله البنائي .

اما في مرحلة التقليد فانه يمكن للقارئ أن يستخدم نفس المنهج المحدد في هذا الكتاب لبناء نظم مشابهة . وعندما يتمكن القارئ من كل مفاتيح عملية تخطيط البرامج يستطيع اجراء التعديلات المختلفة على الامثلة المحددة في هذا الكتاب بما يتلائم مع احتياجات نظامه وبما يمكنه من الوصول الى الهدف المطلوب . عند هذه المرحلة يكون القارئ قد بنى خبرة جيدة وتميزة في انتاج البرامج باستخدام نظم عائلة DBASE يتمكن بعدها من تطوير العديد من البرامج المشابهة والغير مشابهة للنظم المذكورة في هذا الكتاب ثم يلي ذلك مرحلة الابداع حيث يستطيع القارئ ايجاد أساليب مختلفة تعبر عن شخصيته وتميز أسلوبه وقدرته على الابتكار .

ويتكون الكتاب من ست وعشرين فصلاً مقسمة الى خمسة أجزاء . الجزء الأول منها مراجعة شاملة لنظم برامج عائلة DBASE . أما الجزء الثاني فيوضح نظام معلومات لشنون الطلبة الذي يصلح أن يستخدم في العديد من المدارس وبعض كليات الجامعة بطريقة مبسطة وسهلة . ويعتبر الجزء الثالث أحد البرامج الحاسوبية الشائعة الاستخدام وهو يوضح نظام المخازن . أما الجزء الرابع فهو متعلق بنظام حسابات العملاء وهو أيضاً من البرامج الحاسوبية المستخدمة في العديد من المؤسسات التجارية والصناعية . أما الجزء الخامس فانه يضيف بعض الأساليب المتقدمة لتخطيط البرامج والتي يمكن أن يستفيد منها مخطط البرامج في رفع كفاءته وقدرته على

استخدام نظم برامج عائلة DBASE .

ويعتبر هذا الكتاب أحد المحاولات الجيدة التي يمكن ان توضح للدارس بعض النظم التطبيقية الشائعة الاستخدام . ومن هذا البعد - بالاضافة الى ما سبق ذكره - فان هذا الكتاب يعتبر اضافة حقيقية للمكتبة العربية في مجال تكنولوجيا علوم الحاسب .

ا. د. محمد فهمي طلبه

محتسويات الكتاب

رقم الصفحة	الموضوع	مسلسل
١	الفصل الأول : "مقدمة"	
٤	مشيرة النقطة	١ - ١
٤	تعديل مواصفات النظام (Config.sys)	٢ - ١
٧	الجزء الأول : مراجعة كتب (نظم ادارة قواعد البيانات)	
٩	الفصل الثاني : " أنسواع البرامج"	
١١	التعامل مع قاعدة بيانات واحدة	١٠ - ٢
١٢	قواعد البيانات المرتبطة (Relational Databases)	٢ - ٢
١٤	نظام الملف الرئيسي وملفات الحركة	٢ - ٢
١٧	استخدام حقول الملاحظات	٤ - ٢
٢٢	الفصل الثالث : "البحث السريع"	
٢٥	استخدام الفهرس	١ - ٢
٢٦	طرق البحث	٢ - ٢
٢٧	البحث عن مدى معين	٢ - ٢
٢٩	التنفيذ السريع للعمليات الحسابية	٤ - ٢
٣٠	زيادة سرعة طباعة التقارير	٥ - ٢
٣٠	النسخ السريع للسجلات	٦ - ٢
٣١	التعامل مع ملفات فهرس متعددة	٧ - ٢
٣٢	الفصل الرابع : "خطوات تصميم النظام"	
٣٥	تعريف المشكلة (Problem Definition)	١ - ٤
٣٦	توصيف المدخلات والمخرجات (Input/Output)	٢ - ٤
٣٦	تصميم قاعدة البيانات	٣ - ٤
٣٧	التصميم الجزأ للبيانات (Modular Design)	٤ - ٤

مسلسل الموضوع رقم الصفحة

٢٩ الفصل الخامس : كتابة البرامج

٤١	إنشاء ملفات الأوامر	١ - ٥
٤٢	التفاعل مع المستخدم	٢ - ٥
٤٣	الأمر (ACCEPT)	١ - ٢ - ٥
٤٤	الأمر (INPUT)	٢ - ٢ - ٥
٤٤	الأمر (WAIT)	٢ - ٢ - ٥
٤٥	الأمر (@ .. SAY .. GET)	٤ - ٢ - ٥
٤٦	الحلقة التكرارية	٢ - ٥
٤٧	اتخاذ القرار بواسطة الأمر (IF)	٤ - ٥
٤٩	اتخاذ القرار بواسطة الأمر (DO CASE)	٥ - ٥
٥١	الكتابة التركيبية للبرامج	٦ - ٥

٥٥ الفصل السادس : وسائل التصحيح " (Debugging) "

٥٧	مقدمة	١ - ٦
٥٨	عرض الذاكرة (Memory)	٢ - ٦
٥٩	عرض التاريخ (History)	٢ - ٦
٥٩	استخدام الأمر (SET TALK ON)	٤ - ٦
٦٠	استخدام الأمر (SET ECHO ON)	٥ - ٦
٦٠	استخدام الأمر (SET STEP ON)	٦ - ٦
٦٠	استخدام الأمر (SET DEBUG ON)	٧ - ٦
٦١	أهم أخطاء كتابة البرامج	٨ - ٦
٦١	الرسالة (Data type mismatch)	١ - ٨ - ٦
٦١	الرسالة (Invalid function argument)	٢ - ٨ - ٦
٦٢	الرسالة (Unrecognized command verb)	٢ - ٨ - ٦
٦٢	الرسالة (Variable not found)	٤ - ٨ - ٦
٦٢	الرسالة (Record out of range)	٥ - ٨ - ٦
٦٢	الرسالة (Too many files open)	٦ - ٨ - ٦

٦٥ الجزء الثاني : نظام معلومات شئون الطلبة

٦٧ الفصل السابع : تصميم النظام

٦٩	مقدمة	١ - ٧
٦٩	تصميم القائمة الرئيسية	٢ - ٧
٧٠	إضافة أسماء وعناوين جديدة	١ - ٢ - ٧
٧٠	طباعة التقارير والعناوين المختصرة	٢ - ٢ - ٧
٧٢	تعديل البيانات	٢ - ٢ - ٧
٧٢	مسح السجلات	٤ - ٢ - ٧

رقم الصفحة	الموضوع	مسلسل
٧٤	٧ - ٢ - ٥ الخروج من النظام	
٧٤	٢ - ٧ انشاء ملف قاعدة البيانات	
٧٦	٤ - ٧ انشاء شاشة الادخال	
٧٧	٥ - ٧ انشاء التقرير	
٧٧	٦ - ٧ تركيب البرنامج	
٧٩	الفصل الثامن : "البرنامج الرئيسي"	
٨٧	الفصل التاسع : "برنامج التقارير"	
٩٦	١ - ٩ البرنامج (Rep)	
٩٩	٢ - ٩ البرنامج (Label)	
١٠١	الفصل العاشر : " برنامج التصحيح "	
١٠٩	الفصل الحادي عشر : "برنامج مسح السجلات"	
١١٩	الجزء الثالث : نظام المتاجز	
١٢١	الفصل الثاني عشر : "توصيف النظام"	
١٢٢	١ - ١٢ تصميم النظام	
١٢٤	٢ - ١٢ حقول المفتاح	
١٢٥	٢ - ١٢ وظائف النظام	
١٢٥	٤ - ١٢ تحديد حقول الملفات	
١٢٧	٥ - ١٢ تصميم قاعدة البيانات	
١٢٧	١ - ٥ - ١٢ انشاء الملف الرئيسي (Master File)	
١٢٨	٢ - ٥ - ١٢ انشاء ملف المبيعات (Sales File)	
١٢٩	٣ - ٥ - ١٢ انشاء ملف الأصناف الواردة	
١٣٠	٦ - ١٢ تصميم البرنامج	
١٣٢	الفصل الثالث عشر : "برنامج القائمة الرئيسية"	
١٣٥	١ - ١٢ كتابة الخطوات الأولية (PSEUDOCODE)	
١٣٥	٢ - ١٢ كتابة البرنامج	
١٣٩	٣ - ١٢ اختبار البرنامج	

رقم الصفحة	الموضوع	مسلسل
١٤١	الفصل الرابع عشر : "برنامج تشغيل الملف الرئيسي"	
١٤٣	تصميم برنامج تشغيل الملف الرئيسي	١ - ١٤
١٤٤	تصميم البرنامج الرئيسي	٢ - ١٤
١٤٥	برنامج اضافة الأصناف	٣ - ١٤
١٤٦	انشاء شاشة الادخال	١ - ٣ - ١٤
١٤٧	كتابة الخطوات الأولية (PSEUDOCODE)	٢ - ٣ - ١٤
١٤٨	كتابة برنامج الاضافة (Addnumbs.prg)	٣ - ٣ - ١٤
١٥١	برنامج تقارير الملف الرئيسي	٤ - ١٤
١٥١	تقرير المخزون الحالي (Current Stock)	١ - ٤ - ١٤
١٥٢	تقرير حد الطلب (Reorder)	٢ - ٤ - ١٤
١٥٤	تقرير الأصناف تحت الطلب	٣ - ٤ - ١٤
١٥٦	طلب الشراء (Purchase Order)	٤ - ٤ - ١٤
١٥٧	تصميم برنامج التقارير	٥ - ٤ - ١٤
١٥٨	تصميم برنامج القائمة	٦ - ٤ - ١٤
١٦٢	برنامج أوامر الشراء	٧ - ٤ - ١٤
١٦٤	كتابة البرنامج	٨ - ٤ - ١٤
١٧٠	برنامج تعديل الملف الرئيسي	٥ - ١٤
١٧٥	الفصل الخامس عشر : "برنامج تشغيل ملف المبيعات"	
١٧٧	تركيب برنامج المبيعات	١ - ١٥
١٧٨	برنامج القائمة الرئيسية	٢ - ١٥
١٧٩	برنامج نقطة البيع	٣ - ١٥
١٨٣	كتابة الخطوات الأولية (PSEUDOCODE)	١ - ٣ - ١٥
١٨٤	كتابة البرنامج	٢ - ٣ - ١٥
١٩٦	ادخال السعر اكيا	٣ - ٣ - ١٥
١٩٧	برنامج تقارير البيع	٤ - ٥
١٩٩	كتابة الخطوات الأولية للبرنامج	١ - ٤ - ١٥
١٩٩	كتابة البرنامج —	٢ - ٤ - ١٥
٢٠٧	الفصل السادس عشر : "برنامج تشغيل ملف الاضافة"	
٢١٠	تركيب البرنامج	١ - ١٦
٢١١	برنامج قائمة الاضافة (NMenu.prg)	٢ - ١٦
٢١٢	برنامج ادخال بيانات الأصناف (Newstock.prg)	٣ - ١٦
٢١٣	كتابة الخطوات الأولية (PSEUDOCODE)	٤ - ١٦
٢١٣	كتابة البرنامج	٥ - ١٦
٢١٧	برنامج تقارير الاضافة (NewReps.prg)	٦ - ١٦
٢٢٠	كتابة البرنامج	٧ - ١٦

رقم الصفحة	الموضوع	مستلم
٢٢٢	الفصل السابع عشر : "برنامج تحديث البيانات"	
٢٢٥	برنامج تحديث الملف الرئيسي (Master.dbf)	١ - ١٧
٢٢٦	كتابة الخطوات الأولية (PSEUDOCODE)	١ - ١ - ١٧
٢٢٧	كتابة البرنامج	٢ - ١ - ١٧
٢٢٢	برنامج تصحيح ملف المبيعات (SalEdit.prg)	٢ - ١٧
٢٢٤	الخطوات الأولية (PSEUDOCODE)	١ - ٢ - ١٧
٢٢٥	كتابة البرنامج	٢ - ٢ - ١٧
٢٤٢	برنامج تصحيح ملف الاضافة (NewEd.prg)	٢ - ١٧
٢٥٢	الجزء الرابع : نظام حسابات العملاء	
٢٥٥	الفصل الثامن عشر : "تصميم النظام"	
٢٥٧	تعريف المشكلة	١ - ١٨
٢٥٧	تحديد هيكل قاعدة البيانات	٢ - ١٨
٢٥٨	ملف بيانات العميل (Customer.dbf)	١ - ٢ - ١٨
٢٥٩	ملف حركة الصرف (Charges.dbf)	٢ - ٢ - ١٨
٢٦٠	ملف السداد (Payments.dbf)	٢ - ٢ - ١٨
٢٦١	حفظ البيانات التاريخية	٢ - ١٨
٢٦٢	تركيب البرنامج	٤ - ١٨
٢٦٥	الفصل التاسع عشر : "ملفات الخطوات"	
٢٦٧	استخدام ملف الخطوات في برنامج حسابات العملاء (A/R)	١ - ١٩
٢٦٧	برنامج العنوان (Title)	٢ - ١٩
٢٦٩	برنامج رسائل الأخطاء	٢ - ١٩
٢٦٩	برنامج التحقق من رقم العميل	٤ - ١٩
٢٧٠	إنشاء ملف الخطوات	٥ - ١٩
٢٧٢	فتح ملف الخطوات	٦ - ١٩
٢٧٤	ادخال المعاملات (Parameters)	٧ - ١٩
٢٧٥	دراسة برنامج الخطوات (GetCust)	٨ - ١٩
٢٧٩	الفصل العشرون : "برنامج القائمة الرئيسية والإدخال والتعديل"	
٢٨١	برنامج القائمة الرئيسية	١ - ٢٠
٢٨٤	برنامج اضافة العملاء (NewCust.prg)	٢ - ٢٠
٢٨٨	برنامج اضافة حركة الصرف (NewChrg.prg)	٢ - ٢٠
٢٩١	برنامج اضافة حركة السداد (NewPay.prg)	٤ - ٢٠
٢٩٤	برنامج التعديل (AREdit.prg)	٥ - ٢٠
٢٩٥	تعديل ملف العميل (EdCust.prg)	١ - ٥ - ٢٠

رقم الصفحة	الموضوع	مستند
٢٩٧	(EdChrg.prg) تعديل ملف المصرف	٢٠ - ٥ - ٢
٣٠١	(EdPay.prg) تعديل ملف السداد	٢٠ - ٥ - ٢
٣٠٥	الفصل الحادي والعشرون : "تقارير برنامج حسابات العملاء"	
٣٠٧	برنامج قائمة التقارير الرئيسية (ARPrint.prg)	٢١ - ١
٣٠٩	ملف الخطوات (BillProc.prg)	٢١ - ٢
٣١٣	برنامج الفواتير الشهرية (Bills.prg)	٢١ - ٣
٣١٦	برنامج اختبار الحالة (ARStat.prg)	٢١ - ٤
٣١٧	برنامج التقارير المختصرة والزمنية (Agesumm.prg)	٢١ - ٥
٣٢٢	التقارير التاريخية (ARHist.prg)	٢١ - ٦
٣٢٩	الفصل الثاني والعشرون : "التحديث الشهري للنظام"	
٣٢٩	الفصل الثالث والعشرون : "برنامج التكامل بين حسابات العملاء والمخازن"	
٣٤٥	الجزء الخامس : بعض الأدوات المتقدمة	
٣٤٩	الفصل الرابع والعشرون : "برنامج كتابة الشيكات"	
٣٥٩	الفصل الخامس والعشرون : "برنامج اختيار الألوان"	
٣٦٥	الفصل السادس والعشرون : "برنامج تحريك العمود الضوئي"	

الفصل الأول

مقدمة

هذا الكتاب يمثل التطبيق العملي للكتاب السابق على نظم المعلومات الشائعة الاستخدام مثل نظم معلومات شئون الطلبة (Cadets)، ونظم المخازن (Inventory) ونظم حسابات العملاء (Accounts Receivable). وهو يركز على أحدث الوسائل (Techniques) التي تستخدم في كتابة البرامج القوية ذات الكفاءة العالية وسرعة التشغيل الكبيرة .

والكتاب لا يكتفي بعرض البرامج ولكنه يشرح كل برنامج خطوة خطوة حتى يستوعب القارئ البرامج . كما أنه يراعى استخدام معظم أوامر عائلة (DBase) والدوال الخاصة بها واستخدام كل البدائل الممكنة حتى يصبح القارئ ملماً بجميع إمكانيات البرنامج ، كما تتوفر لديه القدرة على كتابة برامج كاملة قابلة للتنفيذ . كما توفر مؤسسة دلتا أقراصاً تحتوي على هذه البرامج ، يستطيع القارئ - الذي لا يجد وقتاً كافياً لكتابتها - استخدامها مباشرة من الأقراص مع إمكانية تتبع سطور البرنامج ومراجعتها قبل تنفيذها .

وهذا الكتاب كما سبق الإيضاح هو استكمال للكتاب السابق ، لذلك فمن المفيد قراءة الكتاب الأول حتى يستطيع القارئ استيعاب البرامج الموجودة . ويستطيع القارئ كتابة هذه البرامج على الحاسب وتنفيذها حيث أن هذا يكسبه الخبرة المطلوبة لكتابة أى برامج أخرى .

والكتاب ينقسم الى خمسة أجزاء ، الجزء الأول عبارة عن مراجعة شاملة لبرامج عائلة (DBase) والأوامر المستخدمة فيها . مع شرح لطرق كتابة البرامج وبعض الوسائل التي تزيد من كفاءتها وسرعة تنفيذها .

والجزء الثاني يشرح برنامج نظام معلومات شئون الطلبة (Cadets) الذي يمثل دوماً من البرامج التي تتعامل مع قاعدة بيانات واحدة (Single Database) . وهو برنامج يصلح للمبتدئين حيث أنه يركز على أساسيات كتابة البرامج التي تعتمد على القوائم في تشغيلها بواسطة المستخدم (User Friendly) . كما يركز أيضاً على تصميم شاشات الإدخال وتصميم التقارير .

والجزء الثالث يشرح برنامج المخازن (Inventory) الذي يوضح أساسيات التعامل مع عدة ملفات قواعد بيانات ويربط هذه الملفات وفتحها من مناطق عمل مختلفة (Work Areas) .

والجزء الرابع يشرح برنامج حسابات العملاء (Accounts Receivable) الذي يضيف إمكانيات متقدمة للتعامل مع الملفات المرتبطة (Related) وتحديثها واستخدام الملفات التاريخية (History Files) للاحتفاظ بالبيانات القديمة .

والجزء الخامس يضيف بعض الوسائل المتقدمة (Advanced Techniques) التي يمكن لمخطط البرامج استخدامها مع أى برنامج لزيادة كفاءته .

ويلاحظ أن الكتاب يتدرج فى درجة صعوبة البرامج حتى يصل بالقارئ الى نهاية الكتاب الى الخبرة الكافية والقدرة على التعامل مع أعقد نظم المعلومات .

ملاحظة

البرامج المشروحة فى هذا الكتاب تعمل على جميع برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro) . كما ان الأوامر المستخدمة فى كتابة هذه البرامج هى نفس الأوامر المستخدمة فى برامج عائلة (DBase) الأخرى . ارجع الى الكتابين الأول والثانى لمراجعة أوامر عائلة (DBase) .

١ - ١ مشيرة النقطة (Dot Prompt)

كما سبق الايضاح فى الكتابين السابقين فان برنامج (DBase III+) ينقسم الى جزأين رئيسيين هما برنامج المساعد (Assistant) ومشيرة النقطة (Dot Prompt) . واستخدام مشيرة النقطة هو الأساس فى كتابة البرامج .

ويتم عرض مشيرة النقطة عن طريق الضغط على مفتاح الهروب (ESC) عند بدء تشغيل برنامج (DBase III+) وظهور قوائم المساعد (Assistant) . ويمكن عرض مشيرة النقطة مباشرة بعد تحميل البرنامج . ولتنفيذ ذلك يتم اجراء تعديل بسيط فى ملف المواصفات (Config.DB) وذلك بإلغاء السطرين التاليين من الملف .

```
STATUS ON
COMMAND = ASSIST
```

١- ٢ تعديل مواصفات النظام (Config.sys)

كما سيلاحظ القارئ فيما بعد ، أن بعض البرامج التى سيتم تصميمها خلال هذا الكتاب تحتوى على العديد من الملفات التى يلزم فتحها فى نفس الوقت ، فإذا

كان ملف مواصفات النظام (Config.sys) لا يحتوى على الأمر الذى يسمح باستخدام عدد كبير من الملفات ، فان من المتوقع عند تشغيل أى برنامج كبير ظهور الرسالة التالية :

Too many files are open

وللتغلب على ذلك يتم تعديل ملف مواصفات النظام (Config.sys) باستخدام برنامج (EDLIN) أو أى برنامج معالجة كلمات مع الشكل غير الوثائقى (Non Document Format) . ويتم كتابة السطرين التاليين .

```
FILES = 20
BUFFERS = 15
```

ويجب بعد ذلك اطفاء الجهاز وإعادة تشغيله من جديد حتى يتم تحميل ملف المواصفات وتخصيص الملفات (Files) ومخازن الذاكرة (Buffers) المطلوبة للبرنامج .



الجزء الأول

مراجعة كتاب (نظم إدارة قواعد البيانات)

- أنواع البرامج
- البحث السريع
- خطوات تصميم النظام
- كتابة البرامج
- وسائل التصحيح

الفصل الثاني

أنواع البرامج

من المهم قبل البدء في تصميم البرنامج ، تحديد نوع هذا البرنامج ، إذا كان من البرامج التي تتعامل مع ملف قاعدة بيانات واحد (Single Database) مثل برنامج شئون الطلبة (Cadets) كما سيتم الايضاح ، أو من البرامج التي تتعامل مع عدة ملفات مرتبطة (Relational Databases) ، أو من البرامج التي تتعامل مع ملف رئيسي وملفات حركة (Master/Transaction Databases) . لذلك سوف يتم القاء الضوء على كل نوع من هذه الأنواع الثلاثة قبل تصميم البرامج التي تنتمي الى كل نوع .

٢ - ١ التعامل مع قاعدة بيانات واحدة

البرنامج في هذه الحالة يعتبر أبسط أنواع البرامج . حيث يكفي تصميم ملف قاعدة بيانات واحد ، ويتعامل البرنامج دائما مع هذا الملف . لذلك لا تكون هناك حاجة الى فتح عدة ملفات واستخدام عدة مناطق عمل (Work Areas) في نفس الوقت . وهذا يسهل التحكم في البرنامج ومتابعة تنفيذه .

وعادة يتم انشاء ملف قاعدة البيانات قبل البدء في كتابة البرنامج . ويتم ذلك بكتابة السطر التالي من مشيرة النقطة (Dot Prompt) .

CREATE Cadets

حيث يكون الملف (Cadets.dbf) هو ملف قاعدة البيانات المطلوب انشاؤه .

وفي هذه الحالة تظهر قائمة تحديد الحقول التي يتم من خلالها تحديد اسم كل حقل ونوعه وعرضه وعدد الأرقام العشرية ان وجدت .

كما يمكن انشاء ملف الفهرس من مشيرة النقطة أيضا كالآتي مثلا :

USE Cadets

INDEX ON Name TO Name

في هذه الحالة يتم ترتيب السجلات بناء على الترتيب الهجائي لأسماء الطلبة . كما يمكن فهرسة الملفات بناء على حقلي ، حيث يكون الحقل الأول هو الحقل الرئيسي الذي يتم الترتيب بناء عليه . وذلك كالآتي مثلا :

USE Cadets

INDEX ON Name + Class TO Name

وعندما يزداد فتح ملف الفهرس يتم كتابة الآتي مثلا :

USE Cadets INDEX Name

وهذا يؤدي الى فتح ملف قاعدة البيانات وملف الفهرس الخاص به . كما يمكن فتح ملف الفهرس وحده عندما يكون قد سبق فتح ملف قاعدة البيانات وذلك كالآتي :

SET INDEX TO Name

٢ - ٢ قواعد البيانات المرتبطة (Relational Databases)

قواعد البيانات المرتبطة هي الملفات التي تكون مرتبطة ببعضها بناء على حقل مشترك . وهي تفيد بصفة خاصة في التخلص من أى تكرار للبيانات . وهذا يؤدي الى تقليل المساحة التخزينية المستخدمة بالإضافة الى زيادة سرعة التشغيل .

فمثلا عند تصميم ملف قاعدة بيانات لحسابات العملاء (AR.dbf) ، يمكن تصميمه كالآتي :

Structure for database :C: AR.dbf

Field	Field Name	Type	Width	Dec
1	BILL_DATE	Date	8	
2	AMOUNT	Numeric	9	2
3	VENDOR	Character	20	
4	ADDRESS	Character	20	

وهذا يعتبر تصميمًا جيدًا للملف ، حيث قد يكون هناك بائعون (Vendors) لهم مئات الفواتير بتاريخ مختلفة . وفي هذه الحالة يتم تكرار أسماء هؤلاء البائعين وعناوينهم .

ولكن التصميم الأفضل هو تقسيم الملف الى ملفين منفصلين ، الملف الأول هو الملف (AR1.dbf) مع تخزين البيانات الشخصية للبائعين مثل الاسم والعنوان في

ملف آخر (AR2.dbf).

فمثلا يمكن انشاء الملف (AR1) كالآتي :

Structure for database :C: AR1.dbf

Field	Field Name	Type	Width	Dec
1	BILL_DATE	Date	8	
2	AMOUNT	Numeric	9	2
3	VEND_CODE	Character	5	

كما يمكن انشاء الملف (AR2.dbf) كالآتي :

Field	Field name	Type	Width	Dec
1	VEND_CODE	Character	5	
2	VENDOR	Character	20	
3	ADDRESS	Character	20	

ويلاحظ أن الحقل الوحيد المكرر هو حقل (Vend_Code) . وهو الحقل الذي يستخدم في ربط الملفين .

ولربط هذين الملفين يتم فهرسة أحدهما على الحقل المشترك (Vend_Code) كالآتي :

```
USE AR2
INDEX ON Vend_Code TO Vendor
```

ثم يتم فتح كل ملف في منطقة عمل مختلفة (Work Area) . ويتم انشاء العلاقة بين الملفين باستخدام الأمر (SET RELATION TO) . وذلك يتضح من السطور التالية :

```
SELECT 1
USE AR1
SELECT 2
USE AR2 INDEX Vendor
SELECT 1
SET RELATION TO Vend_Code INTO AR2
```

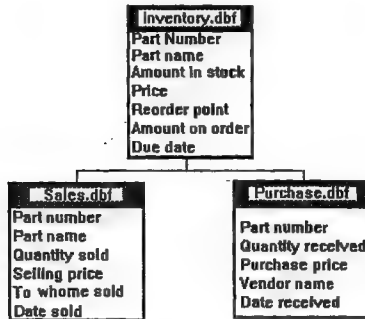
وعندما يراد عرض أى حقل من حقل الملف (AR2) يستخدم الرمز (>- B) . حيث أن (B) هنا يمثل الاسم المرادف (Alias) لمنطقة العمل رقم (2) ، وسيتم شرح ذلك بالتفصيل عند شرح نظام حسابات العملاء (Accounts Receivable).

٢ - ٣ نظام الملف الرئيسى وملفات الحركة

هذا النوع من البرامج يستخدم عادة مع نظم المخازن (Inventory) والمكتبات (Libraries) والبنوك (Banks) ، حيث يكون هناك ملف بيانات رئيسى (Master File) يحتوى على بيانات كل حركة اضافة أو صرف . ويتم تحديث بيانات الملف الرئيسى (Master) من ملفات الحركة (Transaction). لمثلا نظام المخازن في أبسط صورة يحتوى على ملف رئيسى واحد وملفين حركة . ويكون هيكل النظام كما هو موضح من الشكل (٢ - ١) .

والملف الرئيسى في هذا النظام مثلاً يحتوى على البيانات الحالية عن المخزون الفعلى (On hand stock) . كما يحصل على بيانات كل حركة صرف أو توريد من ملفات الحركة حتى يعطى دائماً الموقف الحالى للمخزون . وفائدة هذا النظام أنه يعطى المستخدم الموقف الحالى فى كل لحظة ، كما يتابع حركة الصرف والتوريد .

ويلاحظ وجود حقل مشترك بين الملف الرئيسى وملفات الحركة وهو حقل رقم الجزء (Part_No) . وهو حقل منفرد (Unique) يستخدم فى ربط الملفات الثلاثة . ويستخدم الأمر (UPDATE) فى تحديث الملف الرئيسى من ملفات الحركة . ولتنفيذ ذلك يجب أولاً فهرسة الملفات على حقل رقم الجزء (Part_No) .



شكل (٢ - ١)

فمثلا لتحديث الملف الرئيسي (Master) من ملف المبيعات (Sales) يجب
أولا كتابة السطور التالية :

```

USE Inventory
INDEX ON Part_No TO Master
USE Sales
INDEX ON Part_No TO Sales

```

ثم يتم فتح كل ملف في منطقة عمل مختلفة (Work Area) باستخدام الأمر
(SELECT) ، ثم يستخدم الأمر (UPDATE) في نقل البيانات من ملف المبيعات
الى الملف الرئيسي . وذلك كالآتي :

```

SELECT 2
USE Sales INDEX Sales
SELECT 1
USE Inventory INDEX Master
UPDATE ON Part_No FROM Sales REPLACE On_Hand WITH ;
On_Hand - B -> Qty

```

ملاحظة

يجب ملاحظة أنه عند زيادة الأمر عن طول السطر على الشاشة يتم استخدام حرف الفاصلة المنقوطة (؛) . ويجب أن يعرف مخطط البرامج وظيفة الفاصلة المنقوطة في هذه الحالة حتى لا يحدث خطأ في كتابة الأمر، حيث أن هذا الحرف يؤدي إلى ضم السطر التالي إلى السطر الجارى كتابته . ولذلك يرامى عندما يكون مطلوباً وجود مسافة خالية (Space) ترك هذه المسافة قبل كتابة هذا الحرف . لأن هذا الحرف يضم السطر التالي دون أى مسافات .

ولاحظ هذا في السطر الأخير من الأوامر السابقة . وهذا السطر يؤدي إلى تحديث الملف الرئيسى (Master.dbf) من ملف المبيعات (Sales.dbf) عن طريق استبدال محتويات حقل الكمية الفعلية (On_Hand) بنفس المحتويات مطروحا منها كمية الصنف التى تم بيعها (B -> Qty) والتى يتم الحصول عليها من ملف المبيعات في منطقة العمل رقم (٢) .

ويمكن اجراء عملية التحديث من ملف المشتريات (Purchases) بنفس الطريقة . والفارق الوحيد هو اضافة الكمية المشتراة (B -> Qty) من ملف المشتريات في منطقة العمل رقم (٢) الى الكمية الفعلية . وذلك كالآتى :

```
UPDATE ON Part_No FROM Purchase REPLACE On_Hand WITH ;
On_Hand + B -> Qty
```

وسوف يتم شرح كل الوسائل المتاحة لكتابة البرامج الخاصة بهذا النظام في الجزء الخاص بدراسة برنامج المخازن (Inventory) .

٢ - ٤ استخدام حقول الملاحظات

هناك بعض قواعد البيانات التى تحتاج الى استخدام حقل للملاحظات . وأوضح مثل لذلك قاعدة بيانات المكتبة (Library) . حيث يراد مثلا انشاء ملف يحتوى على بيانات بالابحاث أو الكتب الخاصة بكل مؤلف . هذا الملف يمكن أن يحتوى على حقول اسم المؤلف ، عنوان البحث أو الكتاب ، تاريخ النشر ، اسم الناشر ، الموضوعات (Topics) ، والملخص (Abstract) . ويمكن تصميم هيكل الملف (Structure) في هذه الحالة بطريقتين . في الطريقة الأولى نسمى الملف مثلا (Lib1.dbf) ودخل الحقول الخاصة به كالآتى :

Structure for database :C: Lib1.dbf				
Field	Field name	Type	Width	Dec.
1	AUTHOR	Character	20	
2	TITLE	Character	20	
3	PUB	Character	20	
4	DATE	Date	8	
5	TOPICS	Character	60	
6	ABSTRACT	Character	254	

شكل (٢ - ٢)

ويلاحظ في هذه الحالة أنه تم تخصيص (٢٥٤) حرفاً للملخص (Abstract). وذلك لأن هذا العدد يمثل الحد الأقصى لعرض الحقل الحرفي . وهذا العرض يسمح بكتابة سطور معدودة لا تزيد عن أربعة سطور .

ولكن قد يكون مطلوباً كتابة ملخص كبير يزيد عن صفحة لكل كتاب مثلاً . وفي هذه الحالة يتم استخدام نوع آخر من الحقول يسمى حقل الملاحظات (memo field). وهذا النوع من الحقول يسمح بكتابة حتى (٤٠٠٠) حرفاً في الحقل الواحد . ويمكن زيادة عدد الحروف أكثر من ذلك عن طريق استخدام معالج كلمات (Word Processor) آخر غير المستخدم مع برنامج (DBase III+).

والشكل التالي يوضح هيكل الملف (Lib2.dbf) بعد استخدام حقل الملاحظات (memo field) .

Structure for database Lib2.dbf				
Field	Field name	Type	Width	Dec
1	AUTHOR	Character	20	
2	TITLE	Character	20	
3	PUB	Character	20	
4	DATE	Date	8	
5	TOPICS	Character	60	
6	ABSTRACT	Memo	10	

شكل (٢ - ٣)

ولاحظ هنا أن البرنامج يخصص الرقم (١٠) ألبا لمرض حقل الملاحظات (Abstract) رغم أنه يسمح فعليا بتخزين حتى (٤٠٠٠) حرف . وهذا لأن ما يكتب في هذا الحقل يخزن فعليا في ملف قاعدة بيانات مساعد (Auxiliary) يكون امتداده (.dbt) وليس (.dbf) . ولكن مكان هذا الحقل فقط هو الذي يتم تخزينه في ملف قاعدة البيانات الأصلي .

ولادخال بيانات في حقل الملاحظات يتم ذلك من خلال شاشة الادخال حيث يتم وضع المؤشر على حقل الملاحظات والضغط على مفتاحي (Ctrl-PgDn) . وفي هذه الحالة يتم مسح الشاشة وتصبح الشاشة جاهزة لادخال بيانات هذا الحقل . وعند الانتهاء من الكتابة يتم الضغط على مفتاحي (Ctrl-PgUp) أو مفتاحي (Ctrl-w) أو مفتاحي (Ctrl-end) لتخزين بيانات هذا الحقل والرجوع الى شاشة الادخال مرة ثانية .

ولاحظ عند استخدام الأمر (LIST) في عرض بيانات الحقول عدم ظهور محتويات حقل الملاحظات ولكن تظهر كلمة (memo) . ولكن عندما يرد عرض محتويات هذا الحقل يتم تحديد اسم الحقل مع الأمر (LIST) كالآتي مثلا :

LIST OFF Author,Title,Pub,Date,Abstract

وفي هذه الحالة يظهر الآتي :

Garton J. T. Decision Support Systems Byte Magazine
03/01/85

This article discusses automated Decissions ,
Support Systems used in modern business Mini
and Microcomputer systems. It includes a
review and comparison of several currently
available systems , both as support ,
knowledge Maker , Mind games and Decisive.

Franklin B.W. Automated MBO MBO Monthly
05/01/85

Describes several automated systems that
support Management by objectives (MBO). The
basic theory of MBO is discussed, the several
MBO systems are reviewed.

شكل (٢ - ٤)

ويمكن التحكم في عرض الكتابة في حقل الملاحظات عن طريق كتابة الأمر
(SET MEMOWIDTH) . فمثلا لتحديد العرض (٤٠) لحقل الملاحظات يتم كتابة
الأمر التالي :

SET MEMOWIDTH TO 40

كما يمكن التحكم في مكان ظهور الملاحظات وشكلها من خلال ملف الأوامر
(Command File) . فمثلا لعرض نفس الملاحظات السابقة بطريقة أوضح يمكن
كتابة البرنامج التالي :

```
*****Library.prg
* - - Sample Program to print database with memo field
USE Lib2
GO TOP
```

```
DO WHILE .NOT. EOF()
  ? "AUTHOR      :", Author
  ? "Title       :", Title
  ? "Publisher   :", Pub
  ? "Date        :", Date
  ? "Keywords    :", Topics
  ?
  ? Abstract
  ?
  ?
  SKIP
ENDDO(while not eof)
```

وعند تنفيذ هذا البرنامج تظهر البيانات الواضحة من الشكل (٢ - ٥) .

وهناك تصور واحد في استخدام حقول للملاحظات ، وهو أنه لا يمكن البحث عن الملاحظات التي تحتوي على كلمة معينة مثلا أو موضوع معين . فمثلا عند استخدام الأمر التالي :

```
LIST FOR "Computer" $ Abstract
```

للبحث عن السجلات التي تحتوي على كلمة (Computer) في حقل الملاحظات (Abstract) . في هذه الحالة يلاحظ ظهور الرسالة التالية :

Operation with memo field invalid

ولعلاج هذه المشكلة يمكن استخدام حقل آخر للبحث يسمى (Keywords) كما في المثال السابق ، حيث يتم وضع بعض رؤوس المواضيع (Topics) التي يمكن بعد ذلك البحث من خلالها .

فمثلا عند كتابة الأمر التالي :

```
LIST FOR "Computer" $ Keywords
```

يتم عرض بيانات جميع السجلات التي تحتوي على كلمة (Computer) في حقل الملاحظات .

Record no. : 1
Author : Garton , J ,
Title : Decision Support Systems
Publisher : Byte Magazine
Date : 03/01/89
Key words : Support, knowledge Maker , Mind games

This article discusses automated Decision
Support systems used in modern business.
Mini and micro-----

Record no. : 2
Author : Franklin B. W.
Title : Automated MBO
Publisher : MBO Monthly
Date : 05/01/89
Key words : Management , MBO , Microcomputer

Describes several automated systems -----

الفصل الثالث

البحث السريع

هذا الفصل يركز على الوسائل المختلفة (Techniques) التي يمكن استخدامها في كتابة البرامج لزيادة سرعة التشغيل بدرجة كبيرة . ومن أهم العمليات التي تؤثر بدرجة كبيرة في سرعة تنفيذ البرنامج عملية البحث عن بيان معين خلال ملف قاعدة البيانات ، حيث أن ذلك قد يستغرق أيا ما في الملفات الكبيرة إذا لم يتم كتابة البرامج بالصورة السليمة . أما عند استخدام الوسائل المختلفة التي سيتم شرحها في هذا الباب فقد يصل زمن البحث الى دقائق معدودة وربما ثوان مهما كبر حجم ملف قاعدة البيانات . وأول هذه الوسائل هو استخدام الفهرس (Index) في ترتيب سجلات الملف .

٣ - ١ استخدام الفهرس

أقرب وأوضح مثال لتأثير الفهرس على سرعة البحث عن البيانات هو استخدام فهرس الكتاب . نفرض مثلاً أننا نقرأ في كتاب عن الحاسبات ونريد أن نبحث عن موضوع نظم الخبرة ، فهناك طريقتان للبحث ، الأولى عن طريق فرز صفحات الكتاب صفحة صفحة حتى نصل الى الصفحة التي تحتوي على هذا الموضوع ، وإذا كان الكتاب كبيراً فإن البحث قد يستغرق مدة طويلة . والطريقة الثانية هي الذهاب مباشرة الى الفهرس الموجود في آخر الكتاب والبحث في الفهرس عن كلمة (نظم الخبرة) وتحديد رقم الصفحة المناظر ثم الذهاب الى هذه الصفحة . وعملية البحث في فهرس الكتاب لن تأخذ وقتاً كبيراً لأن هذا الفهرس يكون مرتباً بالترتيب الهجائي .

وما يحدث مع برامج عائلة (DBase) هو نفس الشيء تقريباً ، حيث تكون هناك طريقتان للبحث عن بيان معين في ملف قاعدة البيانات . الطريقة الأولى عن طريق قراءة كل سجل من سجلات الملف للوصول الى السجل الذي يحتوي على البيان المطلوب البحث عنه . والطريقة الثانية هي انشاء ملف فهرس (Index) بناء على الحقل الذي يحتوي على البيان المطلوب البحث عنه ، مثل الاسم ، والبحث عن البيان المطلوب خلال هذا الفهرس ، وتحديد رقم السجل المقابل له ثم الذهاب الى هذا السجل .

ولسي برامج عائلة (DBase) يتم انشاء الفهرس باستخدام الأمر (INDEX ON) . كما يتم تشغيل هذا الفهرس بكتابة اسمه عند فتح ملف قاعدة البيانات كالآتي مثلا :

USE Cadets INDEX Name

كما يتم استخدام الأمر (FIND) أو الأمر (SEEK) في البحث عن البيان المطلوب .

وعند اجراء أى عمليات على سجلات ملف قاعدة البيانات مثل اضافة سجلات جديدة أو مسح سجلات أو تعديل بيانات سجل معين ، يجب مراعاة فتح ملف الفهرس أولا قبل اجراء هذه العمليات ، حيث أن فتح ملف الفهرس يؤدي الى ادخال أى تعديل يتم على ملف قاعدة البيانات على هذا الفهرس .

٣ - ٢ طرق البحث

بالاضافة الى السرعة التي يوفرها استخدام الفهرس في ترتيب السجلات فان هناك طرقا مختلفة للبحث عن السجلات توفرها برامج عائلة (DBase) وتتفاوت سرعة البحث من خلالها . لذلك فمن المهم عرض طرق البحث المختلفة ومقارنة سرعة البحث في كل طريقة حتى يختار مخطط البرامج الطريقة المناسبة التي تزيد من كفاءة وسرعة تنفيذ البرنامج .

فمثلا نفرض أن هناك ملف قاعدة بيانات اسمه (Test.dbf) يحتوى على ألف سجل . ونفرض أن هناك عشرة سجلات تحتوى على الاسم (Mohamed) مثلا . ويراد عرض بيانات هذه السجلات على الشاشة . في هذه الحالة نقوم بمقارنة طريقتين مختلفتين لتنفيذ المطلوب .

في الطريقة الأولى يتم استخدام الأمر (LIST) مع كلمة (FOR) لادخال شرط البحث . والسطور التالية توضح ذلك :

```
CLEAR
USE Test INDEX Name
ACCEPT "List what name ? " TO Search
LIST FOR Name = Search
```

وعند تنفيذ هذا البرنامج يتم مسح الشاشة ويظهر السؤال التالي :

List what name ?

وعند ادخال الاسم (Mohamed) مثلا يتم تخزينه في متغير الذاكرة الحرفي (Search) ، وتظهر بيانات السجلات العشرة التي تحتوى على هذا الاسم . والوقت الذي يستهلك في عرض هذه السجلات يزيد عن الدقيقتين بقليل . وذلك في حالة استخدام الأقراص المرنة (Floppy Disks) . كما يستهلك حوالى ٢٢ ثانية عند استخدام القرص الصلب (Hard Disk) .

والطريقة الثانية لتنفيذ نفس هذه العملية هي استخدام الأمر (FIND) أو الأمر (SEEK) في تحديد رقم أول سجل يحتوى على الاسم (Mohamed) ثم استخدام (WHILE) لعرض باقي السجلات . والسطور التالية توضح ذلك :

```
CLEAR
USE Test INDEX Name
ACCEPT "List what name? " TO Search
SEEK Search
LIST WHILE Name = Search
```

وفي هذه الحالة يتم عرض بيانات السجلات العشرة في خمس ثوان في حالة استخدام الأقراص المرنبة (Floppy Disks) ويتم ذلك في أربع ثوان في حالة استخدام القرص الصلب (Hard Disk) .

٣ - ٣ البحث عن مدى معين

عند البحث عن مدى معين من السجلات محصور بين قيمة معينة لأحد الحقول وقيمة أخرى لهذا الحقل ، فإن ذلك يمكن أن يتم باستخدام الأمر (LIST) مع كلمة (FOR) . كما يمكن استخدام الفهرس مع استخدام الأمر (SEEK) و (WHILE) لتنفيذ نفس العملية . ولتوضيح الفرق بين الحالتين سنقوم بدراسة المثال التالي:

نفرض أنه يراد البحث عن السجلات التي تبدأ من تاريخ معين في حقل التاريخ (Date) وتنتهي بتاريخ آخر . في هذه الحالة نبدأ باستخدام الأمر (LIST) مع كلمة (FOR) كالآتي :

```
USE Test
CLEAR
STORE " " TO Start , Finish
@ 10,2 SAY "Enter start date" GET Start;
PICT "99/99/99"
@ 12,2 SAY "Enter ending date" GET Finish;
PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
STORE CTOD(Finish) TO Finish
LIST FOR Date >= Start .AND. Date <= Finish
```

في هذه الحالة يتم عرض بيانات السجلات المحصورة بين التاريخين الذين يكتبهما المستخدم .

وعند استخدام الطريقة الأخرى يتم كتابة السطور التالية :

```
USE Test INDEX Dates
CLEAR
STORE " " TO Start , Finish
@ 10,2 SAY "Enter start date";
GET Start PICT "99/99/99"
@ 12,2 SAY "Enter ending date";
GET Finish PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
STORE CTOD(Finish) TO Finish
SEEK Start
LIST WHILE Date <= Finish
```

وفي هذه الحالة يتم عرض السجلات أسرع كثيراً من الطريقة الأولى، حيث أن البرنامج يبحث أولاً من السجل المحتوى على تاريخ البداية (Start) . وحيث أن الملف مفهرس على حقل التاريخ (Dates) ، فإن البحث يتم عن السجلات التي تبدأ من السجل الذي تم تحديده بواسطة الأمر (SEEK) وتنتهي بالتاريخ الموجود في المتغير (Finish) . ومع أن هذه الطريقة أسرع كثيراً من الطريقة الأولى إلا أنها تنطوي على شيء من الخطورة . حيث أن الأمر (SEEK) إذا لم يجد السجل المحتوى على تاريخ البداية (Start) فإن البرنامج لا يعرض أي سجلات . لأن مؤشر السجلات (Record Pointer) ينتقل إلى آخر الملف (End of File) . ولذلك يفضل في هذه الحالة استخدام طريقة أخرى تجمع بين الطريقتين للاحتفاظ بسرعة تنفيذ البرنامج . ويتم توضيح ذلك من السطور التالية :

```
USE Test INDEX Dates
CLEAR
STORE " " TO Start , Finish
@ 10,2 SAY "Enter start date";
GET Start PICT "99/99/99"
@ 12,2 SAY "Enter ending date";
GET Finish PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
```

```
STORE CTOD(Finish) TO Finish
SEEK Start
IF FOUND()
    LIST WHILE Date <= Finish
ELSE
    LIST FOR Date >= Satart .AND. Date <= Finish
ENDIF(not found)
```

وهناك طريقة أخرى لتنفيذ نفس الطلب عن طريق استخدام الأمر (SET FILTER TO) وذلك كالآتي مثلا :

```
SET FILTER TO DATE >= Start .AND. DATE <= Finish
LIST
```

٣-٤ التنفيذ السريع للعمليات الحسابية

هناك بعض الأوامر الحسابية مثل (COUNT) ، (SUM) ، (AVERAGE) . وهذه الأوامر يمكن أيضا تقليل وقت تنفيذها للعمليات الحسابية بدرجة كبيرة . فمثلا لحساب عدد السجلات التي تحتوي على الاسم (Mahmoud) يمكن كتابة السطرين التاليين :

```
USE Test
COUNT FOR Name = "Mahmoud"
```

وهذه الطريقة تستهلك حوالي (١٥) ثانية للوصول الى النتيجة المطلوبة ، وهي وجود عشرة سجلات تحتوي على هذا الاسم .

ويمكن تقليل هذا الوقت بدرجة كبيرة جدا بكتابة الأوامر التالية :

```
USE Test INDEX Name
FIND Mahmoud
COUNT WHILE Name = "Mahmoud"
```

في هذه الحالة يتم تنفيذ للطلب في حوالي ثانية واحدة .

ونفس هذه الطريقة يمكن استخدامها مع الأمر (SUM) والأمر (AVERAGE) .

٣ - ٥ زيادة سرعة طباعة التقارير

يمكن استخدام طريقتين أيضا في طباعة التقارير . ولتوضيح الفرق بينهما ، يتم كتابة السطرين التاليين الذين يمثلان الطريقة الأولى .

```
USE Test INDEX Name  
REPORT FORM Rep1 FOR Name ="Mahmoud"
```

وهذه الطريقة تستغرق حوالى ٢٠ ثانية في عرض التقرير على الشاشة . والسطور التالية توضح الطريقة الثانية :

```
USE Test INDEX Name  
FIND Mahmoud  
REPORT FORM Rep1 WHILE Name = "Mahmoud"
```

وهذه الطريقة تستغرق حوالى ٦ ثوان في عرض التقرير على الشاشة .

٣ - ٦ النسخ السريع للسجلات

عندما يراد نسخ مجموعة من السجلات في ملف مؤقت (Temporary) ، يمكن تنفيذ ذلك بطريقتين :

الطريقة الأولى يتم توضيحها من السطرين التاليين :

```
USE Test INDEX Name  
COPY TO Temp FOR Name = "Mahmoud"
```

وهذه العملية تستهلك حوالى ٢٠ ثانية على القرص الصلب (Hard Disk) .

والطريقة الثانية يتم توضيحها من السطور التالية :

```
USE Test INDEX Name  
FIND Mahmoud  
COPY TO Temp WHILE Name = "Mahmoud"
```

هذه الطريقة تؤدي الى تقليل زمن التنفيذ الى مايقرب من اثنتين .

٣ - ٧ التعامل مع ملفات فهرس متعددة

يحتاج مخطط البرامج الى ترتيب السجلات ترتيبا مختلفا حسب العملية المطلوب اجراؤها ، في هذه الحالة يمكنه استخدام عدة ملفات فهرس . وبرنامج عائلة (DBase) تتيح لمخطط البرامج انشاء أى عدد من السجلات ، ولكنه لا يستطيع فتح أكثر من سبعة ملفات فهرس فى نفس الوقت .

فمثلا فى برنامج شئون الطلبة (Cadets) يمكن انشاء فهرس بناء على حقل الاسم (Name) كالاتى :

```
USE Cadets
INDEX ON Name TO Name
```

كما يمكن انشاء فهرس بناء على رقم الفرق مثلا (Class) كالاتى :

```
USE Cadets
INDEX ON Class TO Class
```

ويمكن فتح الفهرسين معا بكتابة السطر التالى :

```
USE Cadets INDEX Name , Class
```

وترتيب كتابة ملفات الفهرس مهم جدا فى هذه الحالة ، حيث أن الملف الأول (Name) يصبح الفهرس الرئيسى (Primary) . فعند استخدام أى أمر من أوامر التعامل مع السجلات مثل (LIST) ، (REPORT) ، فإن البرنامج يعرض السجلات مرتبة حسب الفهرس الرئيسى . وإذا تساوت بيانات بعض السجلات فى حقل الفهرس الرئيسى يتم ترتيبها بناء على الفهرس الثانى وهكذا . كما أن الأوامر (SEEK) ، (FIND) تبحث خلال الفهرس الرئيسى فقط .

وإذا أريد تعديل ترتيب ملفات الفهرس يمكن كتابة السطر التالى مثلا :

```
USE Cadets INDEX Class , Name
```

في هذه الحالة يصبح الملف الأول (Class.ndx) هو الفهرس الرئيسي .
ويرامى دائما فتح جميع ملفات الفهرس التى سبق انشاؤها ثم تعديل ترتيبها بعد
ذلك حسب الحاجة . وذلك لان اى تعديل يحدث فى السجلات مثل اضافة سجلات
جديدة أو مسح سجلات أو تعديل سجلات يؤدي الى تحديث ملفات الفهرس
المفتوحة . أما ملفات الفهرس غير المفتوحة فلا يتم تحديثها وبالتالي تصبح غير
مطابقة للوضع الحالى للسجلات فى ملف قاعدة البيانات . ولعلاج ذلك يتم اعادة
انشاء الفهرس من جديد . ويتم ذلك عن طريق فتح ملفات الفهرس التى سبق
انشاؤها ثم استخدام الأمر (REINDEX) وذلك كالآتى :

```
USE Cadets INDEX Name , Class  
REINDEX
```


الفصل الرابع

خطوات تصميم النظام

عادة يبدأ تصميم النظام بمجرد فكرة . ثم تنمو هذه الفكرة تدريجيا حتى تنتج النظام الكامل . وعادة يبدأ مخطط البرنامج بأن يسأل نفسه (من أين أبدأ ؟) . وعندما يجيب على هذا السؤال ويبدأ في التنفيذ ، فإنه يسأل نفسه بعد كل خطوة (أين أذهب بعد هذه الخطوة) . ومن مجموع هذه الخطوات يصل في النهاية الى التصميم النهائي للنظام .

وخطوات التصميم بصفة عامة يمكن شرحها كالآتي :

- ١ - تعريف المشكلة أو الهدف من النظام .
- ٢ - توصيف المدخلات والمخرجات .
- ٣ - تصميم هيكل قاعدة البيانات (Database Structure) .
- ٤ - تقسيم البرنامج الى برامج فرعية (Modules) يؤدي كل منها وظيفة محددة .
- ٥ - كتابة البرامج الفرعية .
- ٦ - اختبار وتصحيح البرامج .

وسوف يتم شرح كل خطوة من هذه الخطوات في هذا الباب .

٤ - ١ تعريف المشكلة (Problem Definition)

أول خطوة في تصميم النظام هي تحديد المشكلة المطلوب حلها أو الهدف العام للنظام . وكلما كان هذا الهدف محددا وواضحا كان تنفيذ الخطوات التالية أسهل . وأول خطوة في هذا التحديد تبدأ من اسم النظام نفسه مثل (نظام معلومات شئون الطلبة) . وهذا يعني أن المطلوب انشاء نظام يتيح الحصول على معلومات معينة عن الطلبة . ولكن هذا الاسم وحده يكون غامضا بعض الشيء . لذلك يتم توضيحه قليلا عن طريق معرفة خصائص المستخدم النهائي لهذا النظام . فإذا كان المطلوب انشاء نظام معلومات يتيح للمستخدم العادي الذي ليس له أى خبرة بالحاسب استخدام النظام واسترجاع المعلومات المطلوبة ، في هذه الحالة يصبح من السهل على مخطط البرامج تحليل هذا الهدف الى خطوات محددة مثل الآتي :

- ١- اضافة بيانات طلبة جدد .
- ٢- طباعة تقارير متضمنة بيانات طالب معين أو مجموعة من الطلبة .
- ٣ - تعديل بيانات أى طالب .
- ٤ - مسح بيانات أى طالب .

كما يمكن تحليل هذه الخطوات الى خطوات أخرى أكثر تحديدا .

٤ - ٢ توصيف المدخلات والمخرجات (Input/Output)

الخطوة الثانية في تصميم النظام هي تحديد ما يجب ادخاله الى الحاسب (Input) وما يجب أن نحصل عليه منه (Output) . ولإيهم في هذه المرحلة كيفية الادخال أو الاخراج . فمثلا في نظام معلومات شئون الطلبة (Cadets) نريد الحصول على الآتي من الحاسب .

- ١ - تقارير بيانات الطلبة (Reports) تحتوي على الاسم والعنوان ورقم التليفون ... الخ .
- ٢ - تقارير مختصرة للطلبة (Labels) تحتوي على الاسم والعنوان .

وعادة يتم تحديد المدخلات من طريق هذه المخرجات . لذلك نستطيع تحديد المدخلات الآتية :

Name
Nationality
Address
Phone number

٤ - ٣ تصميم قاعدة البيانات

الخطوة الثالثة هي تصميم قاعدة البيانات المطلوبة . وفي هذه المرحلة يجب تحديد نوع كل حقل من حقول ملف قاعدة البيانات . وهنا يجب التمييز بين الحقول العددية (Numeric) وبين الحقول الحرفية (Character) التي تحتوي على أعداد . حيث أن المقصود بالحقول العددية عادة هو الأعداد التي لا يتم استخدامها في عمليات حسابية فمثلا رقم التليفون يتم ادخاله كحقل حرفي وليس عدديا .

ويجب في هذه المرحلة أيضا انشاء ملف الفهرس . ويجب تحديد نوع هذا الملف حسب الحقل المطلوب البحث بناء عليه . فمثلا في نظام معلومات شئون الطلبة يتم كتابة السطرين التاليين لانشاء ملف الفهرس :

USE Cadets
INDEX ON Name TO Name

٤ - ٤ التصميم المجزأ للنظام (Modular Design)

كما سبق الايضاح ، فان أسهل طريقة لتصميم النظام هي تقسيمه الى برامج صغيرة (Modules) كل منها يحقق وظيفة محددة .

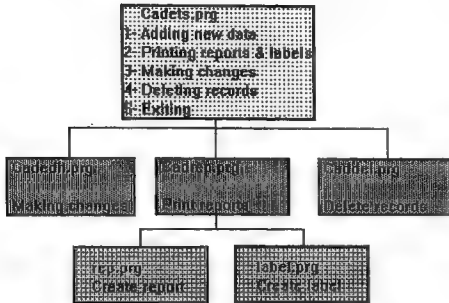
ويستخدم الشكل الهرمي (Hierarchical) في تحديد البرنامج الرئيسي والبرامج الفرعية المتفرعة منه . فمثلا في نظام معلومات شئون الطلبة (Cadets) ، يكون البرنامج الرئيسي هو البرنامج الذى يقوم بعرض القائمة الرئيسية (Main menu) . والبرامج الفرعية هي التي تحقق كل اختيار من اختيارات القائمة . والبرنامج الرئيسى مثلاً يعرض القائمة التالية :

Cadets Information System

- 1 - Add new names and addresses
- 2 - Print reports
- 3 - Make changes
- 4 - Delete names and addresses
- 5 - Exit

Enter choice:

والشكل التالى يوضح التركيب الهرمى للبرنامج :



الفصل الخامس

كتابة البرامج

يتم كتابة البرامج من خلال برنامج (DBase III+) أو برنامج عائلة (DBase) الأخرى من طريق كتابة ملفات الأوامر (Command Files) التي تحتوى على الأوامر المطلوب تنفيذها بالتسلسل المنطقي المطلوب .

ويستخدم الأمر (MODIFY COMMAND) في كتابة ملفات الأوامر . حيث أن هذا الأمر يؤدي إلى تشغيل المصحح الخطي (Text Editor) الخاص ببرنامج (DBase III+) الذي يتم عن طريقه كتابة ملفات الأوامر وتصحيحها . كما يستخدم الأمر (DO) في تشغيل ملفات الأوامر .

٥ - ١ إنشاء ملفات الأوامر

كما سبق الإيضاح فإن لبرنامج (DBase III+) المصحح الخطي الخاص به . وهذا المصحح الخطي يتم تشغيله من طريق الأمر (MODIFY COMMAND) . فمثلا لإنشاء الملف (Test) يتم كتابة الأمر التالي :

MODIFY COMMAND Test

وعند الضغط على مفتاح الإدخال ، تظهر شاشة خالية لكتابة الأوامر خلالها . وتظهر قائمة مساعدة (Help) توضح المفاتيح التي يتم من طريقها التحكم في الكتابة . وهذه المفاتيح يتم شرح وظائفها من خلال الجدول الموضح بالشكل (٥ - ١) .

ويمكن كتابة سطرين من الملف (Test) للتمرين على كتابة ملفات الأوامر كالآتي مثلا :

CLEAR

? "Good morning"

ويتم الضغط على مفتاح الإدخال بعد كتابة كل سطر . وإذا حدث خطأ في الكتابة يتم استخدام مفاتيح التصحيح الموجودة في الجدول في الوصول إلى الحروف المطلوب تعديلها . ثم يتم تخزين الملف من طريق كتابة (^End) أو كتابة (^W) . وهذا يؤدي إلى تخزين الملف (Test) والعودة إلى مشيرة النقطة .

المفتاح	الوظيفة
↑ أو E	تحريك المؤشر سطرا لأعلى
↓ أو X	تحريك المؤشر سطرا لأسفل
S أو <-	تحريك المؤشر حرفا لليسار
^D أو ->	تحريك المؤشر حرفا لليمين
Del أو G	مسح الحرف فوق المؤشر
Ins أو ^V	التحويل إلى حالة الاضافة (Ins) أو الكتابة الفوقية
End أو ^F	تحريك المؤشر كلمة الى اليمين
Home أو ^A	تحريك المؤشر كلمة الى اليسار
^N	اضافة سطر خال مكان المؤشر
^T	مسح كلمة يمين المؤشر
^Y	مسح سطر مكان المؤشر
^End أو ^W	تخزين ملف الأوامر
Esc أو ^Q	الرجوع الى مشيرة النقطة دون تخزين الملف
^Ka	يعيد تشكيل النص (يستخدم عادة مع حقول الملاحظات)
^KF	البحث عن كلمة معينة في النص
^KL	تحديد مكان الكلمة الغائبة التي يتم البحث عنها
^KR	قراءة ملف خارجي في مكان المؤشر
^KW	كتابة ملف معين في ملف آخر باسم آخر

شكل (٥ - ١)

والبرنامج يضيف الامتداد (prg) لاسم الملف . وعندما يراد تشغيل هذا الملف يتم كتابة الآتي :

DO Test

وعند الضغط على مفتاح الادخال يلاحظ مسح الشاشة وظهور الآتي :

Good morning

وبلاحظ أن البرنامج (Test.prg) قام بتنفيذ السطرين ، حيث بدأ بمسح الشاشة (Clear) ثم عرض السطر السابق .

وعندما يراد تعديل هذا البرنامج مثلا يتم كتابة السطر التالي :

MODIFY COMMAND Test

وفي هذه الحالة تظهر السطور السابق كتابتها ويتم تعديلها .

٥ - ٢ التفاعل مع المستخدم

في المثال السابق كان المطلوب فقط عرض رسالة على الشاشة . ولكن في معظم البرامج يكون مطلوبا سؤال المستخدم وانتظار اجابته ، ثم تخزين هذه الاجابة في متغير ذاكرة . وبناء على قيمة هذا المتغير يتم تنفيذ عملية معينة . وهذا يشبه الحديث بين شخصين وتصرف كل منهما بناء على ذلك . وهناك عدة أوامر يتم استخدامها في تحقيق هذا التفاعل (Interaction) بين المستخدم والحاسب . وهذه الأوامر يتم القاء الضوء عليها في الأجزاء التالية .

٥ - ٢ - ١ الأمر (ACCEPT)

يستخدم هذا الأمر في عرض رسالة للمستخدم وانتظار اجابته على هذه الرسالة ، ثم تخزين هذه الاجابة في متغير ذاكرة حرفي . فمثلا يمكن كتابة السطر التالي :

ACCEPT "Send report to printer ? (Y/N)" TO Pr

وعند الضغط على مفتاح الادخال تظهر الرسالة التالية على الشاشة :

Send report to printer ? (Y/N)

وينتظر البرنامج حتى يدخل المستخدم الاجابة ، ثم يقوم بتخزين هذه الاجابة في المتغير (Pr) . وهذا الأمر يفضل استخدامه عندما تكون الاجابة المنتظرة من المستخدم حرفية وليست عددية . وهذا لايعني أنه لايقبل الاجابة العددية : ولكنه يعامل هذه الاجابة كقيمة حرفية حتى لو كانت عددا .

٥ - ٢ - ٢ الأمر (INPUT)

هذا الأمر يشبه الأمر (ACCEPT) حيث يعرض رسالة للمستخدم وينتظر اجابته على هذه الرسالة . ولكن الأمر (INPUT) يتعامل مع اجابة المستخدم حسب نوعها سواء كانت حرفية أو عددية . فإذا كانت الاجابة عددية يقوم بإنشاء متغير عددي . فمثلا يمكن كتابة السطر التالي :

INPUT "Enter your age" TO Age

وهذا يؤدي الى ظهور الرسالة التالية على الشاشة :

Enter your age

وينتظر البرنامج من المستخدم ادخال قيمة معينة . وعلى حسب نوع هذه القيمة يقوم بإنشاء متغير ذاكرة من نفس النوع .

٥ - ٢ - ٣ الأمر (WAIT)

يستخدم هذا الأمر مثل الأمرين السابقين في عرض رسالة للمستخدم ولكنه ينتظر من المستخدم الضغط على أى مفتاح حتى يستمر تنفيذ البرنامج . ويستخدم عادة عندما يرا د ايقاف تنفيذ البرنامج ايقافا مؤقتا (Pause) حتى يستطيع المستخدم قراءة بيانات معينة على الشاشة . وهذا الأمر يمكن كتابته دون كتابة أى شيء بعده . وهذا يؤدي الى ظهور الرسالة التالية :

Press any key to continue

وهذه هي الرسالة المبدئية (Default) .

كما يمكن كتابة أى رسالة أخرى كالآتي مثلا :

WAIT "Press any key To return to main menu "

كما يمكن استخدامه دون عرض أى رسائل وذلك كالآتي :

WAIT " "

كما يمكن استخدامه في انشاء متغير ذاكرة (Pr) يتم فيه تخزين الحرف الذي يكتبه المستخدم وذلك كالآتي مثلا :

```
WAIT "Send report to printer ? (Y/N)" TO Pr
```

وهذا المتغير الذي يتم انشاؤه يكون متغيرا حرفيا . وإذا أريد استخدام هذا الأمر في استقبال قيمة عددية من المستخدم يتم كتابة الآتي :

```
WAIT "Enter your choice (1-5)" TO choice
Choice = VAL(Choice)
```

وفي هذه الحالة يتم انشاء متغير ذاكرة حرفي (Choice) يحتوى على الرقم الذي يدخله المستخدم ثم يتم تحويل هذا المتغير الى قيمة عددية .

٥- ٢ - ٤ الأمر (@...SAY...GET)

يستخدم هذا الأمر مثل الأوامر السابقة في عرض رسالة للمستخدم وانتظار الإجابة على هذه الرسالة ، ثم تخزين هذه الإجابة في متغير ذاكرة . ولكن هذا الأمر يمتاز بالقدرة على التحكم في مكان ظهور الرسالة على الشاشة عن طريق الإحداثيات التي يتم كتابتها بعد الحرف (@) .

وهذا الأمر يختلف عن الأوامر السابقة في أنه يلزم قبل استخدامه انشاء متغير الذاكرة أولا . كما يستخدم الأمر (READ) بعد ذلك في تخزين إجابة المستخدم في هذا المتغير الذي سبق انشاؤه . فمثلا عندما يراد سؤال المستخدم عن الاختيار المطلوب من الشاشة وتخزين هذا الاختيار في المتغير (Choice) يتم كتابة السطور التالية :

```
Choice = 0
@ 10,5 SAY "Enter choice" GET Choice
READ
```

وهذه الأوامر تبدأ بإنشاء المتغير العددي (Choice) ، ثم عرض الرسالة (Enter Choice) في السطر العاشر والعمود الخامس . ثم ينتظر البرنامج من المستخدم ادخال أى قيمة عددية ، ويقوم بتخزينها في المتغير (Choice) .

٥ - ٣ الحلقة التكرارية

تعد الحلقة التكرارية أحد الأشكال الشائعة الاستخدام في البرامج بصفة عامة وفي برامج مائلة (DBase) بصفة خاصة . ويتم تكييفها في برامج مائلة (DBase) بواسطة الأمر (DO WHILE) . وهي تبدأ عادة بهذا الأمر وتنتهي بالأمر (ENDDO) . ولتوضيح وظيفة الحلقة التكرارية يتم كتابة برنامج اسمه (Count) مثلا . هذا البرنامج يتكون من السطور التالية :

```
CLEAR
SET TALK OFF
STORE 1 TO X
DO WHILE X <= 20
? X
X = X + 1
ENDDO
```

وعند تنفيذ هذا البرنامج يظهر الآتي على الشاشة :

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

ويمكن توضيح تنفيذ البرنامج لهذه الحلقة التكرارية كالآتي :

يبدأ البرنامج بالأمر (CLEAR) الذي يؤدي الى مسح الشاشة . يلي ذلك الأمر (SET TALK OFF) الذي يمنع رسائل البرنامج من الظهور على الشاشة أثناء تنفيذه . ثم يتم انشاء المتغير (X) بواسطة الأمر (STORE) واعطائه القيمة (1). ثم تبدأ الحلقة التكرارية بالأمر (DO WHILE X <= 20) . وهذا الأمر يعني البدء في تنفيذ الحلقة التكرارية التي يستمر تنفيذها طالما كانت قيمة (X) أصغر من أو تساوي (٢٠) . والأمر (X ?) يؤدي الى عرض قيمة (X) على الشاشة . كما يستخدم الأمر (X = X + 1) في زيادة قيمة المتغير (X) بواحد في كل مرة . يتم فيها تنفيذ الحلقة التكرارية . والأمر (ENDDO) يحدد نهاية الحلقة التكرارية .

والأمر (DO WHILE) يستخدم عادة عندما يراد المرور على سجلات ملف قاعدة البيانات وتنفيذ عمليات معينة عليها . وفي هذه الحالة فان هذا الأمر يتم كتابته عادة كالآتي :

DO WHILE .NOT. EOF()

وهذا يعني تنفيذ الحلقة التكرارية طالما لم يتم الوصول الى نهاية الملف (EOF()) . وهي اختصار (END OF FILE) .

٥-٤ اتخاذ القرار بواسطة الأمر (IF)

يستخدم هذا الأمر عندما يراد اتخاذ قرار بناء على معلومات معينة . ويمكن توضيح هذا الأمر عن طريق كتابة برنامج نسيه مثلا (IFTest) ويتم كتابته كالآتي :

MODIFY COMMAND IFTest

وفي هذه الحالة تظهر شاشة الكتابة ويتم كتابة السطور التالية :

```
***** IFTest.prg
* - - - - Program to test the (IF) Command
CLEAR
ACCEPT "Turn on printer ? (Y/N)" TO Pr
* - - - - If answer is yes , then
IF UPPER(Pr) = "Y"
    SET PRINT ON
```

```

? "You chose the printer"
EJECT
SET PRINT OFF
* - - - - - If answer is not yes , then
ELSE
    CLEAR
    ? "You chose the screen"
ENDIF

```

وأول سطرين من هذا البرنامج هما سطران مخصصان للملاحظات مخطط البرامج . لذلك فانهما يبدأان بحرف النجمة (*) . وهذه الملاحظات تفيد في تذكر مخطط البرامج باسم البرنامج ووظيفته . كما أنها تفيد داخل البرنامج في تحديد وظيفة كل مجموعة من الأوامر تؤدي وظيفة محددة . والأمر (CLEAR) يؤدي إلى مسح الشاشة .

والسطر التالي يؤدي إلى عرض السؤال ((Turn on Printer? (Y/N) على الشاشة وانتظار اجابة المستخدم . ثم تخزين هذه الاجابة في المتغير (Pr) .

والسطر التالي يقوم باختبار الحرف الذي يكتبه المستخدم فاذا كان هذا الحرف بعد تحويله إلى حرف كبير (Uppercase) مساويا للحرف (Y) ، يتم تنفيذ مجموعة الأوامر المحصورة بين الأمر (IF) والأمر (ELSE) . وفائدة استخدام الدالة (UPPER) هي الحصول على نفس النتيجة سواء أدخل المستخدم الحرف كبيرا (Uppercase) أو صغيرا (LowerCase) . وإذا أدخل المستخدم أي حرف آخر غير (Y) أو (Y) . يقوم البرنامج بتخطي الأوامر المحصورة بين (IF) ، (ELSE) وينفذ الأوامر المحصورة بين (ELSE) ، (ENDIF) . واختبار هذا البرنامج يتم كتابة هذا الأمر من مشيرة النقطة :

DO IFTest

ولمى هذه الحالة يتم مسح الشاشة ثم تظهر الرسالة التالية :

Turn on Printer ? (Y/N)

فاذا تم كتابته (Y) والضغط على مفتاح الادخال يتم طباعة الرسالة التالية على الطابعة :

You chose the printer.

ثم يتم نقل صفحة على الطابعة (EJECT) .

وإذا تم كتابة أى حرف آخر غير (Y) أو (y) يقوم البرنامج بمسح الشاشة وعرض الرسالة التالية :

You chose the screen.

وتستخدم كلمة (ENDIF) فى انهاء مجموعة الأوامر الخاصة بالأمر (IF) . ويجب ملاحظة أن كل أمر (IF) له كلمة (ENDIF) خاصة به . أما كلمة (ELSE) فهي اختيارية .

ويمكن استخدام الدالة (IIF) فى تحقيق نفس العمل الذى يؤديه الأمر (IF) مع توفير فى عدد السطور المطلوب كتابتها . فمثلا يمكن كتابة السطر التالى :

?IIF (X < 10, "Less Than" , "Greater Than")

هذا الأمر يختبر قيمة (X) فإذا كانت أصغر من (10) يتم عرض الرسالة (Less than) على الشاشة ، وإذا كانت أكبر من (10) يتم عرض الرسالة (Greater than) على الشاشة .

ولتنفيذ نفس هذه العملية بواسطة الأمر (IF) يتم كتابة السطور التالية :

```
IF X < 10
    ? "Less Than"
ELSE
    ? "Greater Than"
ENDIF
```

ومن ذلك يلاحظ أن الدالة (IIF) قد وفرت فى عدد السطور .

٥ - ٥ اتخاذ القرار بواسطة الأمر (DO CASE)

الأمر (DO CASE) يعتبر صورة أكثر شمولاً من الأمر (IF) ، حيث أنه يتيح للبرنامج الاختيار بين عدة حالات تبعاً لقيمة متغير معين . ولتوضيح ذلك يمكن كتابة برنامج نسميه مثلاً (CaseTest) . ولتنفيذ ذلك يتم كتابة الأمر التالى من مشيرة النقطة (Dot Prompt) .

MODIFY COMMAND CaseTest

ثم يتم كتابة الأوامر الموضحة بالشكل (٥ - ٢) . وعند الانتهاء يتم تخزين هذا البرنامج بكتابة (^End) .

```

* * * * * CaseTest.prg
* - - - - - Test the DO CASE command
CLEAR
INPUT "Enter a number from 1 to 4" TO X
DO CASE
    CASE X = 1
        ? "You entered one"
    CASE X = 2
        ? "You entered two"
    CASE X = 3
        ? "You entered three"
    CASE X = 4
        ? "You entered four"
    OTHERWISE
        ? "Invalid number"
ENDCASE

```

شكل (٥ - ٢)

والسطر الأول من البرنامج بعد سطرى الملاحظات يؤدي الى مسح الشاشة .
والسطر التالى يؤدي الى عرض الرسالة التالية على الشاشة :

Enter a number from 1 to 4

ثم ينتظر البرنامج حتى يتم ادخال رقم والضغط على مفتاح الادخال . فى هذه الحالة يتم تخزين هذا الرقم فى المتغير العددي (X) .

ثم يبدأ اتخاذ القرار بناء على قيمة (X) . فاذا كانت (X) تساوى (1) يعرض البرنامج الرسالة التالية :

You entered one

ثم يترك باقى الاختيارات .

وإذا كانت (X) تساوى (2) يعرض البرنامج الرسالة التالية :

You entered two

ثم يترك باقى الاختيارات . وهكذا .

أما إذا تم ادخال رقم لايقىق أيا من هذه الاختيارات ، يتم تنفيذ الأمر التالى لكلمة (OTHERWISE) . وهو يؤدى الى ظهور الرسالة التالية :

Invalid number

٥ - ٦ الكتابة التركيبية للبرامج

من المهم جدا كتابة البرامج بالطريقة التركيبية (Structured) . حيث أن هذه الطريقة تسمح لمخطط البرامج بالرجوع الى البرنامج واختباره وتصحيحه بسهولة كما تسمح لأى مخطط برامج بقراءة البرنامج الذى تمت كتابته بواسطة شخص آخر والقيام بتعديله أو تطويره حسب الحاجة .

ولكتابة البرنامج بهذه الطريقة يجب تنفيذ الآتى :

- ١ - استخدام الملاحظات الواضحة التى تشرح وظيفة كل مجموعة من الأوامر تؤدى وظيفة محددة .
- ٢ - تحريك بدايات السطور (Indent) فى الحلقات التكرارية والسطور الخاصة بالأمر (IF) والأمر (DO CASE) حتى تظهر بداية ونهاية السطور بوضوح .
- ٢ - اختيار الأوامر والمتغيرات التى توضح الوظيفة التى يتم تنفيذها .

ويمكن توضيح فائدة هذه الطريقة عن طريق مقارنة برنامجين أحدهما مكتوب دون مراعاة الطريقة التركيبية (Structured) . انظر الشكل (٥ - ٢) . والآخر يتم فيه مراعاة هذه الطريقة . انظر الشكل (٥ - ٤) .

وفى البرنامج الأول يلاحظ أن قواعد الكتابة التركيبية غير مطبقة . حيث يلاحظ عدم وجود ملاحظات كافية توضح وظيفة كل مجموعة من الأوامر . كما يلاحظ عدم تحريك السطور للداخل (Indentation) فى السطور التى تحتاج الى ذلك . وهذا يؤدى الى عدم وضوح بداية الحلقة التكرارية ونهايتها وكذلك بالنسبة للسطور التالية للأمر (IF) أو الأمر (DO CASE) . وعندما تزيد الحلقات التكرارية كما يحدث فى معظم البرامج تصبح عملية اختبار البرنامج وتصحيحه عملية معقدة جدا .

```

* * * * * Library.prg
* - - - Example of an unstructured program
USE Library
DO WHILE .T.
CLEAR
@ 1,20 SAY "Library Management system"
@ 3,25 SAY "1. Add new records"
@ 4,25 SAY "2. Print Reports"
@ 5,25 SAY "3. Edit data"
@ 6,25 SAY "4. Exit"
STORE 0 TO Choice
@ 8,20 SAY "Enter choice(1-4) " GET Choice
READ IF Choice = 1
APPEND
ELSE
IF Choice = 2
REPORT FORM Library
ELSE
IF Choice = 3
EDIT
IF Choice = 4
RETURN
ENDIF
ENDIF
ENDIF
ENDIF
ENDDO

```

شكل (٥ - ٢)

كما يلاحظ استخدام بعض الأوامر التي قد تعطي معنى مخالفا للواقع مثل الأمر (DO WHILE .T.) . وهذا يعنى أن الحلقة التكرارية سيتم تنفيذها الى ما لانهاية . والحقيقة أنها لا تنفذ الى ما لانهاية ، لأن اختيار المستخدم للرقم (4) يؤدي الى توقف تنفيذها .

والبرنامج بهذه الصورة سوف يتم تنفيذه كما أنه سوف يؤدي الوظيفة المطلوبة .
ولكنه لا يعتبر برنامجاً موثقاً (Documented) يمكن الرجوع اليه وتعديله أو تطويره .

```

* * * * * Library.prg
* - - - - - Library system main menu.
USE Library
STORE 0 TO CHOICE

DO WHILE Choice # 4
    CLEAR "
    @ 1,20 SAY "Library Management system "
    @ 3,25 SAY "1. Add new records "
    @ 4,25 SAY "2. Print Reports "
    @ 5,25 SAY "3. Edit data "
    @ 6,25 SAY "4. Exit
    @ 8,20 SAY "Enter choice(1-4) " GET Choice
    READ
    * - - Branch according to user's request.

    DO CASE
        CASE Choice = 1
            APPEND
        CASE Choice = 2
            REPORT FORM Library
        CASE Choice = 3
            EDIT
    ENDCASE
ENDDO(while choice # 4)

* - - - - - when choice = 4 exit.
RETURN

```

شكل (٥ - ٤)

أما البرنامج الثاني (الشكل (٥ - ٤)) فإنه يؤدي نفس وظيفة البرنامج السابق ولكنه مكتوب بالطريقة التركيبية . حيث يلاحظ أن كل الملاحظات واضحة تماما ومكتوبة بأسلوب واضح . كما يلاحظ أن كل السطور داخل الحلقة التكرارية تم تحريكها للداخل قليلا (Indentation) مما يجعل من السهل تحديد بداية الحلقة التكرارية ونهايتها (يمكن تحريك الأصبع بدءا من الأمر (DO WHILE) رأسيا حتى يصل إلى الأمر (ENDDO) الخاص به) . كما يمكن تحديد بداية الأمر (DO CASE) ونهاية السطور الخاصة به بنفس الطريقة . كما تم استبدال الأوامر التي كانت تسبب شيئا من القموض بأوامر أخرى واضحة . حيث تم استخدام الأمر (DO WHILE Choice # 4) بدلا من الأمر (DO WHILE .T.) .

كما يلاحظ استخدام الملاحظات بجانب كلمة (ENDDO) لتوضيح لمخطط البرامج أي حلقة تكرارية تتبع لها هذه الكلمة . وهذا يفيد بصفة خاصة عندما تزيد الحلقات التكرارية في البرنامج . حيث أن أي ملاحظات تكتب بعد كلمة (ENDDO) أو كلمة (ENDIF) لا يشعر بها البرنامج وتعامل مثل أي ملحوظة مكتوبة بعد الحرف (*) .

الفصل السادس

وسائل التصحيح (Debugging)

٦ - ١ مقدمة

يجب أن يكون واضحاً أنه نادراً ما يوجد البرنامج الذي يكتب أول مرة بدون أخطاء . والأخطاء عموماً قد تكون بسيطة مثل الهجاء الخاطيء للأوامر (Misspelling) . وهذه الأخطاء يسهل اكتشافها عند تشغيل البرنامج . وقد تكون أخطاء منطقية (Logical) وهي عادة تكون معقدة ويصعب اكتشافها . حيث أنها قد لا تؤدي إلى إيقاف البرنامج ولكنها تعطي نتيجة غير النتيجة المتوقعة منه .

وبرامج عائلة (DBase) توفر كثيراً من أدوات التصحيح (Debugging Tools) التي تسهل على مخطط البرامج اكتشاف الأخطاء وتصحيحها .

فمثلاً عندما يجد البرنامج خطأ يؤثر في تنفيذ البرنامج فإنه يعرض السطر الذي يحتوي على الخطأ والبرنامج الذي يحتوي على الخطأ مع الرسالة التحذيرية التالية :

Cancel , Ignore or Suspend? (C,I or S)

وهذه الاختيارات تعني الآتي :

١- الاختيار (cancel)

هذا الاختيار يؤدي إلى إنهاء تشغيل البرنامج والعودة إلى مشيرة النقطة (Dot Prompt) . وفي هذه الحالة تختفي كل متغيرات الذاكرة الخاصة (Private) التي تم أنشاؤها خلال البرنامج .

٢- الاختيار (suspend)

هذا الاختيار يؤدي إلى توقف البرنامج مؤقتاً مع ظهور الرسالة (Do suspend) . وفي هذه الحالة تظل متغيرات الذاكرة الخاصة موجودة في الذاكرة . كما يمكن استكمال البرنامج في أي وقت عن طريق كتابة الأمر (RESUME) من مشيرة النقطة .

٣ - الاختيار (Ignore)

يؤدي هذا الاختيار الى تخطي السطر المحتوى على الخطأ ومحاولة تنفيذ باقي سطور البرنامج اذا لم تكن معتمدة على هذا السطر .

ويمكن ايقاف البرنامج في أى لحظة أثناء تنفيذه بالضغط على مفتاح الهروب (ESC) . وفي هذه الحالة تظهر نفس الاختيارات الثلاثة السابق شرحها .

٦-٢ عرض الذاكرة (Memory)

من الوسائل الفعالة في اكتشاف أخطاء البرنامج ، عرض الذاكرة (Memory) أثناء تنفيذ البرنامج .

فعند حدوث خطأ معين مثلا يمكن كتابة الأمر التالى :

DISPLAY MEMORY

في هذه الحالة يتم عرض حالة الذاكرة المؤقتة (RAM) من حيث متغيرات الذاكرة الموجودة بها وأسمائها ومحتوياتها وأنواعها . ويمكن في هذه الحالة اكتشاف سبب الخطأ عن طريق مراجعة أسماء متغيرات الذاكرة وأنواعها .

وللحصول على محتويات الذاكرة مطبوعة على الورق يتم كتابة السطر التالى :

DISPLAY MEMORY TO PRINT

كما يمكن عرض هيكل ملف قاعدة البيانات المفتوح عن طريق كتابة السطر التالى :

DISPLAY STRUCTURE

في هذه الحالة يمكن اختيار أسماء الحقول وأنواعها واكتشاف أى خطأ موجود . كما يمكن استخدام الأمر (DISPLAY STATUS) لعرض أسماء ملفات قواعد البيانات المفتوحة وكذلك ملفات الفهرس .

٦-٣ عرض التاريخ (History)

المقصود بالتاريخ (History) هو ذاكرة خاصة يتم فيها تخزين آخر عشرين أمراً تم ادخالها من خلال مشيرة النقطة (Dot Prompt). وهذا يؤدي الى امكانية الرجوع الى أى أمر تم ادخاله من آخر عشرين أمراً ، وذلك بالضغط على مفتاح السهم لأعلى (↑) مثلاً . وهذا يوفر على مخطط البرامج كتابة الأمر عدة مرات ، حيث يكفي الضغط على مفتاح السهم لأعلى (↑) عدة مرات حتى يظهر الأمر المطلوب ثم الضغط على مفتاح الإدخال .

ولكن هذا لاينطبق على ملفات الأوامر (Command files) . حيث أن أوامر السلف لا يتم تخزينها في ذاكرة التاريخ (History) الا بعد استخدام الأمر (SET DOHISTORY ON) لتجهيز ذاكرة التاريخ لتخزين الأوامر التي يتم تنفيذها تباعاً .

فمثلاً يمكن كتابة هذا الأمر قبل تنفيذ البرنامج . فإذا توقف البرنامج نتيجة خطأ معين يمكن استخدام الأمر (SUSPEND) في انهاء البرنامج مؤقتاً ، ثم يتم كتابة الأمر (DISPLAY HISTORY) أو الأمر (LIST HISTORY) .

في هذه الحالة يتم عرض آخر عشرين أمراً تم تنفيذها في البرنامج وقد يؤدي هذا الى اكتشاف مكان الخطأ .

ويمكن زيادة عدد الأوامر التي يتم ادخالها في ذاكرة التاريخ الى أكثر من عشرين أمراً من طريق كتابة السطر التالي مثلاً :

SET HISTORY TO 50

ويجب ملاحظة أن الأمر (SET DOHISTORY ON) يؤدي الى ابطاء تنفيذ البرنامج بدرجة كبيرة ، ولذلك يرامى عند الانتهاء من عملية التصحيح (Debugging) اعادة الأمر الى الوضع المبدئى . وذلك بكتابة السطر التالى :

SET DOHISTORY OFF

٦-٤ استخدام الأمر (SET TALK ON)

عادة يتم كتابة الأمر (SET TALK OFF) في أى برنامج حتى لا تظهر

الوسائل التي توضح كل خطوة يتم تنفيذها . وهذه الوسائل تكون مفيدة جدا عند تصحيح البرنامج . ولذلك يفضل أثناء اختيار البرنامج وتصحيحه كتابة الأمر (SET TALK ON) قبل تشغيل البرنامج . كما يتم متابعة الوسائل التي تظهر على الشاشة عند تنفيذ كل خطوة . فعند حدوث خطأ معين في البرنامج ، قد تؤدي هذه الوسائل الى توضيح سبب هذا الخطأ .

٦-٥ استخدام الأمر (SET ECHO ON)

عند كتابة هذا الأمر قبل تنفيذ البرنامج ، فان هذا يؤدي الى عرض كل سطر على الشاشة قبل تنفيذه . ويجب اعادة الأمر الى الوضع المبدئي (Default) عند الانتهاء من عملية التصحيح وذلك بكتابة السطر التالي :

SET ECHO OFF

٦-٦ استخدام الأمر (SET STEP ON)

عند استخدام الأمر (SET ECHO ON) فان الوسائل التي توضح خطوات التنفيذ تظهر على الشاشة أثناء تنفيذ البرنامج كما سبق الايضاح . ولكن الوسائل في هذه الحالة تظهر سريعة مع سرعة تنفيذ هذه الخطوات . فاذا اريد ابطاء ظهور هذه الوسائل حتى يستطيع مخطط البرامج متابعتها فيمكنه في هذه الحالة استخدام الأمر (SET STEP ON) . وهذا يؤدي الى عرض كل أمر عند تنفيذه ، ثم توقف البرنامج مؤقتا ، ثم عرض الرسالة التالية :

Press SPACE to stop, S to suspend, or Esc to cancel

ويمكن في هذه الحالة لمخطط البرامج الضغط على مسطرة المسافات (Space) لتنفيذ الأمر التالي . أو كتابة (S) لتعليق تنفيذ البرنامج ، وذلك عند ملاحظة خطأ معين مثلا ، أو الضغط على مفتاح الهروب (ESC) للخروج من البرنامج . وهذا يتيح لمخطط البرامج التحكم في ظهور رسائل تنفيذ الأوامر أثناء تشغيل البرنامج مما يساعده على اكتشاف الأخطاء المنطقية (Logical Errors) في البرنامج .

٦-٧ استخدام الأمر (SET DEBUG ON)

يستخدم هذا الأمر لارسال خطوات تنفيذ البرنامج الى الطابعة . وهو يستخدم

مع الأمر (SET ECHO ON) والأمر (SET STEP ON) لمتابعة خطوات البرنامج خطوة بخطوة حتى يتم اكتشاف مكان الخطأ .

٦ - ٨ أهم أخطاء كتابة البرامج

في هذا الجزء يتم توضيح أهم رسائل الأخطاء (Error Messages) التي تظهر أثناء تنفيذ البرنامج ، ووسائل علاج هذه الأخطاء .

٦ - ٨ - ١ الرسالة (Data type mismatch)

تظهر هذه الرسالة عادة عند معاملة نوع معين من البيانات كنوع آخر .
مثل معاملة البيانات الحرفية كعددية مثلا ، أو البيانات التاريخية كبيانات
حرفية وهكذا . فمثلا عند كتابة السطر التالي :

LIST FOR Date = "01/30/90"

تظهر الرسالة (Data type mismatch) . وذلك لأن الحقل
(Date) حقل تاريخي يحتوى على تاريخ معين . أما الطرف الأيمن
("01/30/90") فهو بيان حرفي لوجود علامات التنصيص
(Quotation Marks) . ولعلاج ذلك يجب تحويل التاريخ (Date)
إلى القيمة الحرفية المناظرة كالآتي :

LIST FOR DTOC(Date) = "01/30/90"

٦ - ٨ - ٢ الرسالة (Invalid function argument)

تظهر هذه الرسالة عادة عند استخدام نوع معين من البيانات مع دالة
تستخدم نوعا آخر . فمثلا عند كتابة السطر التالي :

? UPPER(X)

وبفرض أن المتغير (X) يحتوى على قيمة عددية (Numeric) . في
هذه الحالة تظهر الرسالة :

Invalid function argument

وذلك لأن الدالة (UPPER) تستخدم فقط لتحويل المدخلات الحرفية الى حروف كبيرة (Uppercase) .

٦- ٨ - ٢ الرسالة (Unrecognized command verb)

تظهر هذه الرسالة عادة عند كتابة هجاء الأمر خطأ ، وفي هذه الحالة يتم مراجعة هجاء الأمر . كما يتم مراجعة هيئة هذا الأمر (Syntax). ويمكن استخدام شاشات المساعدة (Help) في تحديد الهيئة (Syntax) الخاصة بكل أمر .

٦- ٨ - ٤ الرسالة (Variable not found)

تظهر هذه الرسالة عادة عند كتابة اسم متغير ذاكرة لم يسبق تعريفه في البرنامج . وفي هذه الحالة يمكن استخدام الأمر (DISPLAY MEMORY) لعرض محتويات الذاكرة ومعرفة أسماء المتغيرات الموجودة في الذاكرة . كما يمكن استخدام الأمر (DISPLAY STRUCTURE) لعرض أسماء الحقول في ملف قاعدة البيانات المفتوح .

٦- ٨ - ٥ الرسالة (Record out of range)

تظهر هذه الرسالة عند محاولة الذهاب الى سجل (Record) غير موجود في الملف . فمثلا عند كتابة السطر التالي :

GOTO 99

يفرض أن الملف يحتوي على (98) سجلا فقط . في هذه الحالة تظهر الرسالة المذكورة .

وتظهر هذه الرسالة أيضا عند حدوث خطأ في ملف الفهرس المفتوح نتيجة عدم فتحه عند اجراء بعض التعديلات في ملف قاعدة البيانات مثلا . وهذا يتم علاجه بإعادة انشاء الفهرس باستخدام الأمر (REINDEX) .

٦- ٨ - ٦ الرسالة (Too Many Files Open)

تظهر هذه الرسالة عادة عند فتح عدد كبير من الملفات في نفس

الوقت . مع عدم تغيير عدد الملفات في ملف مواصفات النظام (config.sys) ليسمح بفتح هذا العدد من الملفات . في هذه الحالة يتم تعديل ملف المواصفات وتخصيص عدد الملفات (Files) المطلوبة للبرامج . كما يمكن التحكم في البرامج لتقليل عدد الملفات المفتوحة في كل وقت معين . ويتم ذلك عن طريق إغلاق كل ملف بمجرد انتهاء التعامل معه .

كما يمكن استخدام ملف الخطوات (Procedure file) في الاستغناء عن بعض الملفات ، حيث يمكن أن يحتوي ملف الخطوات على عدد من البرامج بحد أقصى ٢٢ برنامجاً بدلاً من كتابة كل منها في ملف منفصل .

2

الجزء الثاني

نظام معلومات شئون الطلبة

- * تصميم النظام
- * البرنامج الرئيسي
- * برنامج مسح السجلات

الفصل السابع

تصميم النظام

٧ - ١ مقدمة

الهدف من هذا البرنامج هو تصميم نظام معلومات للطلبة يمكن السيطرة عليه من خلال القوائم (Menu Driven). وهذا يعنى أن الشخص الذى يقوم بتشغيل البرنامج لا يحتاج الى معرفة أى شيء عن خصائص برنامج (DBase III+) وربما لا يحتاج الى قدر كبير من المعلومات عن الحاسب. ولكن يمكنه عن طريق مجموعة من الاختيارات الواضحة تنفيذ أى عمليات مطلوبة للحصول على أى معلومات أو تعديل البيانات الموجودة أو مسحها أو ... الخ.

٧ - ٢ تصميم القائمة الرئيسية (Main Menu)

يتكون النظام فى هذه الحالة من أربعة برامج يتم ربطها والسيطرة عليها بواسطة برنامج خامس رئيسى. والبرنامج الرئيسى هو البرنامج الذى يقوم بعرض قائمة الاختيارات للمستخدم. وهذه القائمة تعتبر قلب النظام، وهى أول ما يظهر أمام المستخدم وآخر ما يظهر أمامه، كما يتم الرجوع اليها دائما بعد تنفيذ كل مهمة.

ويقوم المستخدم بتشغيل النظام بكتابة الأمر

(DO Cadets)

من مشيرة النقطة. حيث (Cadets) هو اسم البرنامج الرئيسى، وفى هذه الحالة تظهر أمام المستخدم القائمة التالية:

CADETS INFORMATION SYSTEM	
1-	ADD NEW NAMES AND ADDRESSES
2-	PRINT REPORTS OR LABELS
3-	MAKE CHANGES
4-	DELETE NAMES AND ADDRESSES
5-	CHANGE COLOR
6-	EXIT

<p>HILIGHT OPTION BY USING ↑ OR ↓ AND PRESS ← OR PRESS APPROPRIATE MENU NUMBER</p>
--

شكل (٧ - ١)

وسوف يتم شرح كل اختيار من هذه الاختيارات والوظائف التي يؤديها قبل الدخول في تفاصيل البرامج .

٧ - ٢ - ١ اضافة أسماء وعناوين جديدة

وهذا الاختيار يتيح للمستخدم اضافة سجلات جديدة . حيث يقوم المستخدم بكتابة الرقم (1) فتظهر أمامه الشاشة التالية .

CASE NO	1	DATE ENTERED	11/11	TELEPHONE	1111
NAME				BLOOD TYPE	
CLASS		ADDRESS			
RELIGION		NATIONALITY		BIRTH DATE	11/11
BIRTH PLACE		FATHER NAME			
MOTHER NAME				FATHER SALARY	1111
FATHER JOB		NO OF BROTHERS	2	SEC SCHOOL AUG	11/11
DATE OF REC. SCHOOL	11/11	HOBBIES			
NEAREST RELATIVE ADDRESS					

شكل (٧ - ٢)

وهي تشمل الحقول التي يتم ادخال البيانات فيها مع وجود مؤشر صغير على أول حقل . وبعد اضافة البيانات تعود القائمة الرئيسية للظهور مرة أخرى .

٧ - ٢ - ٢ طباعة التقارير والعناوين المختصرة

ويتم تنفيذ هذا الاختيار عندما يكتب المستخدم الرقم (٢) . وفي هذه الحالة تختفي القائمة الرئيسية وتظهر القائمة الموضحة بالشكل (٧ - ٣) .

SELECT A REPORT CHOICE
1 - REPORT 2 - LABELS 3 - RETURN TO MAIN MENU

ENTER YOUR CHOICE(1-3) >

شكل (٧ - ٢)

ومعندما يختار المستخدم أحد هذه الاختيارات ، يظهر السؤال التالي :

Do you want (A)ll records, or (Q)uery?

واختيار (All) عن طريق كتابة الحرف (A) يؤدي الى عرض جميع بيانات الطلبة المخزنة في قاعدة البيانات . واختيار الاستعلام (Query) عن طريق كتابة الحرف (Q) يؤدي الى ظهور الرسالة التالية :

Enter name to display

مع ظهور عمود ضوئي (Highlight) يتم خلاله كتابة الحروف الاولى من الاسم المطلوب ، ويكفي في هذه الحالة كتابة أول حرف من الاسم فقط حيث يقوم البرنامج بعرض جميع الأسماء التي تبدأ بهذا الحرف مع أرقام السجلات الخاصة بها . كما يظهر أمام المستخدم السؤال التالي :

Which one do you want?

وهذا يتيح للمستخدم اختيار الاسم المطلوب عن طريق كتابة رقم السجل الخاص به .

وبعد اختيار المستخدم للاسم المطلوب عرض بياناته يظهر السؤال التالي :

Send report to the printer ? (Y/N)

فإذا كتب المستخدم الحرف (Y) يتم طباعة التقرير . وعند كتابة (N) يتم عرضه على الشاشة فقط . ويظهر في الحالتين التقرير الموضح بالشكل (٤ - ٧) .

CADET NO	:	6960
NAME	:	MOHAMED ALY SALEM
CLASS	:	55P
DATE OF ENTERING AIR ACADEMY	:	09/01/85
BLOOD CLASS	:	A
ADDRESS	:	13 - ABBAS ELAKKAD
TELEPHONE NO	:	2603556
RELIGION	:	MUSLIM
NATIONALITY	:	EGYPTIAN
HOBBIES	:	FOOTBALL
BIRTH DATE	:	01/22/67
BIRTH PLACE	:	TANTA
SECONDARY SCHOOL AVERAGE	:	70.00
DATE OF GETTING SECONDARY SCHOOL	:	1985
FATHER NAME	:	ALY SALEM
MOTHER NAME	:	FATMA MAHMOUD
FATHER SALARY	:	600
NUMBER OF BROTHERS	:	5

شكل (٤ - ٧)

٧ - ٢ - ٢ تعديل البيانات (Make Changes)

عندما يختار المستخدم الاختيار رقم (٢) من القائمة الرئيسية تختفي القائمة الرئيسية ويظهر الآتي على الشاشة :

Enter name of person to edit
or just press Return to Quit:

ويمكن للمستخدم ادخال أول حرف فقط من الاسم كما سبق الايضاح .
وعندما يدخل المستخدم اسما غير موجود في قاعدة البيانات يظهر الآتي على الشاشة :

There is no <name>
Press any key to try again

مع ملاحظة أن الاسم الخطأ الذي يدخله المستخدم يظهر مكان (<name>). وعندما يجد البرنامج هذا الاسم تظهر شاشة البيانات الخاصة به ويظهر المؤشر على أول حقل لتعديله حسب الحاجة .

وعندما يكون هناك أكثر من سجل بنفس الاسم أو مبتدئين بنفس الحرف الذي قام المستخدم بادخاله يقوم البرنامج بعرض الأسماء على المستخدم ومعهما أرقام السجلات حتى يقوم باختيار رقم السجل المقابل للاسم المطلوب .

وبعد أن يقوم المستخدم باجراء التعديل المطلوب على السجل تعود القائمة الرئيسية للظهور من جديد .

٧ - ٢ - ٤ مسح السجلات

عندما يختار المستخدم الاختيار رقم (4) من القائمة الرئيسية ، فإن البرنامج يسأل عن الاسم المراد مسح بياناته . ويقوم المستخدم بادخال الاسم المطلوب أو أول حرف منه فقط كما سبق الايضاح . وفي هذه الحالة تظهر أمام المستخدم الأسماء التي تشترك في هذا الحرف ومعهما أرقام السجلات حتى يقوم باختيار السجل المطلوب . وفي هذه الحالة يتيح البرنامج للمستخدم التأكد من رغبته في مسح هذا السجل عن طريق عرض بيانات السجل أمام المستخدم وسؤاله إذا كان يريد مسح هذا السجل أم لا ، وذلك كالآتي :

Record#	name	address
1	Salem Emam	12-dwawin-Cairo

Delete this record?(Y/N)

وعندما يكتب المستخدم الحرف (Y) يظهر السؤال عن الاسم المراد مسح بياناته مرة أخرى لكي يدخل المستخدم اسما آخر إذا أراد . وعند انتهائه

من ادخال كل الاسماء التي يريد مسحها يمكنه الضغط على مفتاح الادخال بدلا من ادخال اسم جديد . وفي هذه الحالة يتيح له البرنامج التأكد مرة ثانية انه يريد مسح جميع الاسماء التي ادخلها عن طريق عرض الشاشة التالية :

Records to be deleted..

Record#	name	address
6 *	Eman Salem	12 - Dawawin - Cairo
9 *	Medhat Taher	10 - Nasr city - Cairo

Delete all these ? (Y/N)

وعندما يكتب المستخدم (N) . فان البرنامج يتيح له استعادة لى سجل من هذين السجلين حتى لا يتم مسحه نهائيا . وتتكرر هذه العملية حتى يتأكد المستخدم تماما من السجلات التي يريد مسحها نهائيا . وفي هذه الحالة يقوم بكتابة (Y) أمام السؤال (Delete all these?) . فتختفي هذه السجلات تماما من قاعدة البيانات .

٧ - ٢ - ٥ الخروج من النظام (Exit)

عندما يختار المستخدم الاختيار رقم (5) فانه يخرج من البرنامج ويعود الى مشيرة النقطة (Dot Prompt) .

٧ - ٢ - ٦ انشاء ملف قاعدة البيانات

يتم انشاء ملف قاعدة البيانات عن طريق قوائم برنامج المساعد (Assistant) او عن طريق كتابة الأمر (CREATE Cadets) من مشيرة النقطة (Dot Prompt) . وذلك كما سبق الايضاح من خلال الكتاب الأول . حيث تظهر القائمة الموضحة بالشكل (٧ - ٥) والتي يتم من خلالها تحديد أسماء الحقول (Fields) ونوع كل حقل وعرضه وعدد الأرقام العشرية .

CURSOR <--> Char: + - Word: Home End Par: ^_ ^_	INSERT Char: Ins Field: ^N Help: F1	DELETE Char: Del Word: ^Y Field: ^H	Up a field: ↑ Down a field: ↓ Exit/Save: ^End Abort: Esc
---	---	---	---

	Field Name	Type	Width	Dec	Field Name	Type
1	NAME	Character	25			
2	JOB	Character	10			
3	ADDRESS	Character	30			
4	AGE	Numeric	2	0		
5	NOTES	Memo	10			
6	PRICE	Numeric	7	2		
7	QTY	Numeric	7	2		

MODIFY STRUCTURE <<C> <FOS1> <Field: 1/2>

شكل (٧ - ٥)

ونفترض أن هيكل الملف كالآتي :

Field	Field Name	Type	Width	Dec
1	NAME	Character	35	
2	CLASS	Character	4	
3	ADDRESS	Character	40	
4	SEC_SCHOOL	Numeric	5	2
5	DT_ENT	Date	8	
6	T_NO	Character	8	
7	BLOOD	Character	3	
8	RELLIGION	Character	12	
9	NATION	Character	10	
10	B_DATE	Date	8	
11	B_PLACE	Character	12	
12	FATH_NAME	Character	32	
13	MOTH_NAME	Character	33	
14	F_SALARY	Numeric	10	

15	F_JOB	Character	14
16	NO_BROTHER	Numeric	5
17	DATE_SEC	Numeric	5
18	HOBBIES	Character	38
19	N_RELATIVE	Character	35
20	NOTES	Memo	10

ويمكن انشاء ملف الفهرس أيضا عن طريق برنامج المساعد (Assistant) كما سبق الايضاح . أو عن طريق كتابة الأمر التالى من مشيرة النقطة :

INDEX ON UPPER(name) TO Name

ويؤدى هذا الأمر الى انشاء الملف الفهرسى (Name.ndx) .

وتم استخدام الدالة (UPPER) هنا لتحويل الأسماء فى الفهرس الى حروف كبيرة . ويؤدى هذا الى توحيد شكل الأسماء داخل الفهرس بصرف النظر عن شكلها فى ملف قاعدة البيانات . وهذا يؤدى الى سرعة الوصول الى الاسم المطلوب كما سيتم الايضاح فيما بعد .

وبعد انشاء ملف قاعدة البيانات وملف الفهرس الخاص به ، يتم فتح هذين الملفين أو اغلقهما من خلال البرنامج كما سيتم الايضاح . وإذا أراد المستخدم تعديل قاعدة البيانات دون استخدام البرنامج فيجب فى هذه الحالة التأكد من فتح ملف الفهرس مع ملف قاعدة البيانات حتى يتم تحديث الفهرس تبعاً لآى تعديل فى قاعدة البيانات . ويتم ذلك عن طريق كتابة الأمر التالى :

USE Cadets INDEX Name

وهذا يؤدى الى فتح ملف قاعدة البيانات (Cadets.dbf) وملف الفهرس (Name.ndx) فى نفس الوقت .

٧ - ٤ انشاء شاشة الإدخال

يحتاج البرنامج الى تصميم شاشة ادخال (Screen) حتى يستطيع المستخدم عن طريقها ادخال البيانات الى ملف قاعدة البيانات . ويمكن انشاء شاشة الإدخال من

طريق قوائم المساعد (Assistant) كما سبق الايضاح . كما يمكن انشاؤها أيضا بواسطة برنامج يتم التفرع اليه من البرنامج الرئيسي . وكلا الطريقتين سبق شرحهما في الكتاب الأول ، ولذلك سنفترض هنا انشاء شاشة الادخال الموضحة بالشكل (٧ - ٢) ، ونفترض أن الملف الخاص بهذه الشاشة اسمه (Cadets.fmt) .

٧ - ٥ انشاء التقرير

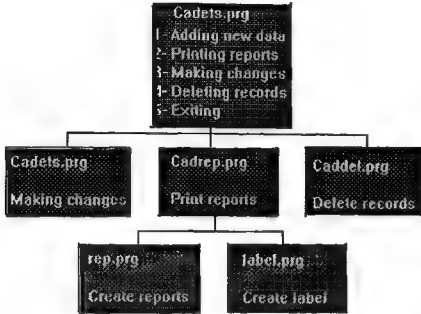
يمكن انشاء التقرير عن طريق قوائم برنامج المساعد (Assistant) أيضا أو من طريق الأمر (CREATE REPORT) . وفي الحالتين تظهر القوائم التي سبق شرحها والتي يتم عن طريقها تحديد شكل التقرير المطلوب طباعته . مع ملاحظة أن انشاء التقرير لايعنى كتابة أي بيانات من ملف قاعدة البيانات فيه ، ولكنه يحدد فقط الحقول المطلوب اظهارها في التقرير ومواقع هذه الحقول وأطوالها ثم تظهر البيانات بعد ذلك عند استخدام هذا التقرير في عرض بيانات سجلات معينة في الملف .

وهناك طريقة أخرى لتصميم التقرير ، وذلك عن طريق برنامج فرعي يتم كتابته . وسوف نستخدم هذه الطريقة في انشاء التقرير لأنها تحقق مرونة كبيرة في توزيع الحقول على الشاشة والتحكم في شكل التقرير . وهذا سيتم شرحه عند دراسة البرامج الفرعية المختلفة .

٧ - ٦ تركيب البرنامج

عند تصميم نظام يحتوي على عدة برامج فمن المهم في هذه الحالة رسم الشكل الهرمي (Hierarchical Structure) لهذه البرامج . ومع أن ذلك قد يبدو غير مطلوب مع هذا النظام ، لأنه لايتحتوى على العديد من البرامج ، فانه من الأفضل دائما التعمد على رسم الشكل الهرمي حتى يمكن تتبع كل برنامج على حدة ومعرفة البرنامج الذي يقوم بالاستدعاء (Calling Program) والبرنامج الذي يتم استدعاؤه (Called Program) .

أنظر الشكل (٧ - ٦) .



شكل (٧ - ٦)

يتضح من هذا الشكل الهرمي أن النظام قد تم تقسيمه الى عدة وظائف رئيسية، وكل وظيفة من هذه الوظائف ينفذها برنامج معين . كما يتضح أيضا أن البرنامج (Cadets.prg) هو البرنامج الذي يقع على قمة الهرم ، وبالتالي فإنه يسيطر على باقي البرامج .

كما أن كل برنامج فرعي بعد انتهاء مهمته يعود الى البرنامج الرئيسي ليقوم بتوزيع باقي المهام .

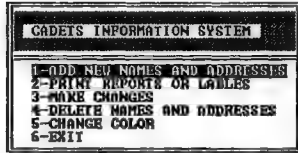
وعملية تقسيم النظام الى وظائف صغيرة يتم تنفيذها من خلال برامج فرعية تسهل تصميم النظام الى درجة كبيرة جدا . حيث يصبح من السهل تصميم وتطوير واختبار كل برنامج صغير على حدة . كما أن ذلك يوفر على مخطط البرامج البحث في عدد كبير من سطور البرنامج عندما يريد تعديل أى جزء من البرنامج أو تصحيحه . حيث يمكنه في هذه الحالة تحديد المهمة أو الوظيفة التي يريد تعديلها ثم يقوم بتعديل البرنامج المسئول عن تنفيذ هذه المهمة .

الفصل الثامن

البرنامج الرئيسي

يفضل دائما قبل البدء فى تصميم البرنامج كتابة الخطوات المطلوبة باللغة الواضحة بالنسبة لمخطط البرامج سواء كانت اللغة الانجليزية أو العربية أو أى لغة أخرى ، وتسمى هذه الخطوات (PSEUDOCODE) . وفى البرنامج الرئيسى الخاص بالطلبة (Cadets) تكون هذه الخطوات كالآتى مثلا :

- ١- تجهيز بيئة البرنامج (Environment) .
- ٢- فتح ملف قاعدة البيانات وملف الفهرس المصاحب له .
- ٣- انشاء حلقة تكرارية لعرض القائمة الرئيسية .
- ٤- مسح الشاشة .
- ٥- عرض القائمة الرئيسية كالآتى :



HILIGHT OPTION BY USING ↑ OR ↓ AND PRESS↵
ON PRESS APPROPRIATE MENU NUMBER

شكل (٨ - ١)

- ٦- استقبال اختيار المستخدم .
- ٧- التفرع الى الأمر أو البرنامج الذى يحقق اختيار المستخدم .
- ٨- إعادة عرض القائمة الرئيسية فى حالة عدم اختيار الخروج (Exit) .
- ٩- الرجوع الى مشيرة النقطة فى حالة اختيار الخروج (Exit) .

وبعد الانتهاء من كتابة الخطوات الأولية (PSEUDOCODE) يتم كتابة البرنامج الذى يحقق هذه الخطوات كالآتى :

```

***** Cadets.prg
***** Cadets Information System

* - - - - - Set up initial parameters
SET TALK OFF
SET STATUS OFF
SET DEFAULT TO C

* - - - - - Open the database and index files
USE CADETS INDEX Name

* - - - - - Begin loop for main menu
choice = 0
DO WHILE choice <> 5
    CLEAR

    TEXT

        Cadets Information System

        1 - Add new names and addresses
        2 - Print Reports
        3 - Make changes
        4 - Delete names and addresses
        5 - Exit

    ENDTEXT

* - - - - - Get users choice
@ 16,20 SAY "Enter choice: " GET choice Picture "9";
RANGE 1,5
READ

* - - - - - Branch to appropriate task or program.
DO CASE
    CASE choice = 1
        SET FORMAT TO Cadets
        APPEND

```

```

CLOSE FORMAT
CASE choice = 2
DO cadrep
CASE choice = 3
DO Cadedit
CASE choice = 4
DO Caddel
ENDCASE
ENDDO (while choice <>5)

* - - - - - Returning to the dot prompt
SET TALK ON
SET STATUS ON
RETURN

```

ويمكن شرح اجزاء البرنامج كالآتي :

يبدأ البرنامج بعرض اسم البرنامج ووظيفته ثم أوامر تجهيز بيئة البرنامج مثل الأمر (SET TALK ON) الذي يمنع ظهور خطوات التشغيل على الشاشة ، ثم الأمر (SET STATUS OFF) الذي يسمح عمود الحالة (Status Bar) من أسفل الشاشة أثناء تنفيذ البرنامج ، ثم الأمر (SET DEFAULT TO C) الذي يجعل القرص الصلب هو القرص المستخدم (Default Drive) . ويمكن تغيير هذا الأمر إذا كان البرنامج موجودا على قرص من باستبدال الحرف (C) بالحرف (A) أو الحرف (B) حسب الحاجة .

والجزء الثاني من البرنامج يفتح ملف قاعدة البيانات (Cadets.dbf) ولف الفهرس المصاحب له (Name.ndx) .

والجزء الثالث يقوم بإنشاء الحلقة التكرارية التي يستمر تنفيذها حتى يكتب المستخدم الرقم (5) للخروج من البرنامج ، ويتم من خلال هذه الحلقة التكرارية عرض القائمة الرئيسية للبرنامج . والأمر (TEXT) والأمر (ENDTEXT) يعرضان جميع السطور المحصورة بينهما والخاصة بقائمة الاختيارات كما سبق الايضاح .

ويلاحظ في بداية الحلقة التكرارية انشاء متغير الذاكرة (choice) واعطائه القيمة صفر .

والجزء الرابع يقوم باستقبال اختيار المستخدم وتخزينه في المتغير (choice) الذي سبق إنشاؤه . ويلاحظ استخدام الأوامر (@, SAY, GET, READ) . كما يلاحظ استخدام عبارة ("Picture=9") . واستخدام الرقم (9) هنا يجبر المستخدم على إدخال أعداد وليس حروف . كما يلاحظ استخدام عبارة (RANGE 1,5) . وهذا يجبر المستخدم على إدخال رقم بين (1,5) لأن أي رقم آخر لن يقبله البرنامج .

والجزء الخامس يستخدم الأمر (DO CASE) الذي يساعد البرنامج على اتخاذ القرار بناء على اختيار المستخدم . وعندما يختار المستخدم الرقم (1) فإن البرنامج يفتح شاشة الإدخال الخاصة بملف قاعدة البيانات المفتوح . وذلك من خلال الأمر :

SET FORMAT TO Cadets

كما يسمح للمستخدم بإضافة سجل جديد عن طريق الأمر (APPEND) . وبعد الانتهاء من إدخال هذا السجل يتم إغلاق ملف شاشة الإدخال عن طريق الأمر (CLOSE FORMAT) . وفي هذه الحالة تظهر القائمة الرئيسية مرة أخرى .

والاختيارات الأخرى (2, 3, 4) تؤدي إلى التفرع إلى برامج فرعية خارجية .

والجزء السادس والأخير من البرنامج يشكل أوامر نهاية الحلقة التكرارية التي يتم تنفيذها في حالة اختيار المستخدم للاختيار (5) . ويلاحظ إضافة تعليق (Comment) بجوار الأمر (ENDDO) يوضح وظيفة الحلقة التكرارية وذلك لأن أي تعليق (Comment) يتم إضافته بعد الأمر (ENDDO) أو الأمر (ENDIF) لا يؤثر في تنفيذ البرنامج . وهذا يكون مفيداً بصفة خاصة عند وجود عدة حلقات تكرارية متداخلة ، حيث يوضح التعليق (Comment) أي (ENDDO) تتبع أي (DO WHILE) .

ثم يلاحظ بعد انتهاء الحلقة التكرارية استخدام الأوامر التي تعيد بيئة الحاسب إلى ما كانت عليه قبل البرنامج . وهي الأوامر التالية :

SET TALK ON
SET STATUS ON
RETURN

ملاحظة

تجدر الاشارة الى أن المستطيلات التي تظهر فى شاشة القائمة الرئيسية و كذلك
الممود المتحرك تتطلب اضافة برنامج فرعى خاص (Routine) . و لتبسيط شرح
البرنامج للمبتدئين فقد تم حذف هذا البرنامج الفرعى . كما تم شرح هذا البرنامج
الفرعى فى نهاية الكتاب للرجوع اليه عندما يراد اضافة أى مؤثرات خاصة فى أى
قائمة .

الفصل التاسع

برنامج التقارير

كما سبق الإيضاح فإن برنامج التقارير هو البرنامج (Cadrep.prg) الذي يتم التفرع اليه من البرنامج الرئيسي في حالة اختيار المستخدم للاختيار رقم (2) . وهذا البرنامج يقوم بعرض قائمة اختيارات فرعية تشمل ثلاثة اختيارات لتحديد نوع التقرير المطلوب أو الرجوع الى القائمة الرئيسية . كما يظهر سؤال عما اذا كان المطلوب عرض التقرير على الشاشة أو طباعته على الطابعة . ويتم توضيح الخطوات الأولية (PSEUDOCODE) كالآتي :

١ - مسح الشاشة وعرض شاشة نوع التقرير المطلوب كالآتي :

SELECT A REPORT CHOICE
1 - REPORT
2 - LABELS
3 - RETURN TO MAIN MENU

ENTER YOUR CHOICE(1-3)]

شكل (١ - ٩)

- ٢ - في حالة عدم اختيار أى نوع يتم الرجوع الى القائمة الرئيسية مرة ثانية .
- ٢ - السؤال عما اذا كان المطلوب عرض جميع بيانات قاعدة البيانات في التقرير أو البحث (Query) عن سجل معين وعرضه .
- ٤ - عند اختيار البحث (Query) يتم سؤال المستخدم عن الاسم المطلوب البحث عنه .
- ٥ - يتم البحث عن السجل المطلوب .
- ٦ - يتم حصر عدد السجلات التي تحتوى على نفس الاسم .
- ٧ - في حالة عدم العثور على أى سجل يحتوى على الاسم المطلوب يتم تنبيه المستخدم حتى يدخل اسما آخر .
- ٨ - في حالة وجود عدة سجلات تحتوى على نفس الاسم يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل .
- ٩ - يتم السؤال عما اذا كان المطلوب عرض التقرير على الشاشة أم طباعته على الطابعة .
- ١٠ - يتم طباعة التقرير .
- ١١ - عند اختيار تقرير (Report) يتم التفرع الى برنامج (Rep.prg) .
- ١٢ - عند اختيار تقرير مختصر (Label) يتم التفرع الى برنامج (Label.prg) .

- ١٢- إيقاف الشاشة مؤقتاً (Pause) في حالة عرض التقرير على الشاشة حتى يستطيع المستخدم قراءة هذا التقرير .
١٤- العودة إلى القائمة الرئيسية مرة ثانية .
- بعد الانتهاء من كتابة الخطوات الأولية (PSEUDOCODE) يتم كتابة البرنامج كالآتي :

```
*****Cadrep.prg
* - - Reports program for Cadets Information system.

* - - - - - Clear screen and ask about report type
CLEAR
TEXT

                Select a report type

                1 - Report
                2 - Label
                3 - Return to main menu

ENDTEXT

* - - - Initialize variable and ask form report type.
repchoice = 0
15,20 SAY "Enter your choice (1-3)" GET repchoice;
        PICTURE "9" RANGE 1,3
READ

* - - - - - If return chosen, return to main menu.
IF repchoice = 3
        RETURN
ENDIF

* - - - - - Ask about query.
CLEAR
qchoice = "A"
@ 10,16 SAY "Do you want (A)ll records,or a (Q)uery?";
        GET qchoice PICTURE "1"
READ
```

```

* - - - - - making Query if requested
SET EXACT OFF
IF qchoice = "Q"
SET EXACT OFF
qname = SPACE(8)
@ 16,16 SAY "Enter name to display" GET qname
READ
qname = UPPER(TRIM(qname))
SEEK qname
recnumb = RECNO()
COUNT WHILE UPPER(name) = qname TO howmany

* - - - - - IF the name is not found, warn the user
IF howmany = 0
    @ 19,15 SAY "There is no & qname"
    ? CHR (7)
    WAIT "Press any key to return to main menu"
    RETURN
ENDIF(howmany = 0)

* - - - - IF there is more than one record having the
* - - - - same name , they are displayed to the user .
IF howmany > 1
    CLEAR
    GO TOP
    SEEK qname
    DISPLAY name, class WHILE UPPER(name) = qname
    @ ROW()+2,15 SAY "which one do you want";
    GET recnumb
    READ
    GOTO recnumb
ELSE
    SET FILTER TO UPPER(name) = qname
ENDIF(howmany>1)
ENDIF(qchoice = Q)

```

```

* - - - - - Ask about printer
toprint = "N"
CLEAR
@ 10,5 SAY "Send records to printer? Y/N" GET toprint;
PICTURE "I"
READ

* - - - - - Displaying the report
IF toprint = "Y"
    SET DEVICE TO PRINT
ELSE
    SET DEVICE TO SCREEN
ENDIF
DO CASE
    CASE repchoice = 1
        DO rep
    CASE repchoice = 2
        DO label
ENDCASE

* - - - - - If printer was not selected ,pause
* - - - - - before returning to menu
IF toprint <> "Y"
    @ 24,1 CLEAR
    WAIT "press any key to return to menu ..."
ENDIF

* - - - - - When report is done, close filter,
* - - - - - and return to main menu
SET FILTER TO
RETURN

```

والبرنامج يبدأ كالعادة بكتابة اسم البرنامج . ثم يبدأ الجزء الأول بمسح الشاشة وعرض قائمة اختيارات نوع التقرير المطلوب . ثم يقوم الجزء الثاني بإنشاء متغير الذاكرة (repchoice) وتخزين القيمة صفر فيه . وهذا يجعل المتغير عددياً (numeric) . ثم يتم عرض رسالة للمستخدم (prompt) لإدخال الاختيار المطلوب وتخزينه في المتغير (repchoice) . ويتم استخدام الصورة (PICTURE"9") لإجبار المستخدم على إدخال أعداد . كما يستخدم المدى (RANGE 1,3) لتحديد مدى الأرقام الذي يمكن إدخاله بحيث لا يزيد عن

(٢) وهذا الجزء يحتوى على السطور التالية :

```
* - - - - Initialize variable and ask for report type
repchoice = 0
15,20 SAY "Enter your choice (1-3)" ;
GET repchoice PICTURE "9" RANGE 1,3
READ
```

ويجب ملاحظة أنه عندما يزيد طول الأمر عن عرض الشاشة يتم كتابة الحرف (;) فى نهاية السطر واستكمال كتابة الأمر فى السطر التالى .

والجزء الثالث من البرنامج يؤدي الى العودة الى القائمة الرئيسية فى حالة كتابة المستخدم الرقم (3) . وهو يحتوى على السطور التالية :

```
IF repchoice = 3
    RETURN
ENDIF
```

والجزء الرابع يسأل المستخدم اذا كان يريد عرض جميع سجلات الملف فى التقرير أو يريد البحث عن سجل معين وعرض بياناته . ويتم تخزين اختيار المستخدم فى المتغير (qchoice) . وهذا الجزء يتكون من السطور التالية :

```
qchoice = "A"
@ 10,16 SAY "Do you want (A)ll records,or a(Q)uery?" ;
GET qchoice PICTURE "1"
READ
```

والجزء الخامس يقوم بالبحث عن الاسم الذى يدخله المستخدم فى حالة اختيار الحرف (Q) . وفى هذه الحالة يتم عرض رسالة للمستخدم لادخال الاسم المطلوب البحث عنه ثم تخزين هذا الاسم فى المتغير (qname) . ويتم تحويل هذا الاسم الى حروف كبيرة من طريق الدالة (UPPER) . كما يتم مسح المسافات الزائدة عن طريق الدالة (TRIM) . ويمكن للمستخدم ادخال حرف واحد أو أكثر من الاسم .

ويتم البحث بواسطة الأمر (SEEK) ثم تخزين رقم السجل الذى يتم الوصول اليه فى المتغير (recnumb) ثم يتم حصر عدد السجلات التى تشترك فى هذا الاسم أو الحرف الذى يتم ادخاله ، وتخزين هذا العدد فى المتغير (howmany) .

ويستخدم الأمر (SET EXACT OFF) حتى تكون المقارنة بين الاسم أو الحرف الذي يتم ادخاله وبين حقل الاسم (name) غير كاملة . أى يكفى وجود الحروف فى أول الاسم لتحقيق شرط البحث . ومع أن الوضع المبدئي (Default) هو (SET EXACT OFF) إلا أنه يفضل كتابته فى البرنامج خشية أن يكون قد تم تغيير وضعه المبدئي قبل تنفيذ البرنامج .

ويتكون هذا الجزء من السطور التالية :

```
IF qchoice = "Q"
    SET EXACT OFF
    qname = SPACE(8)
    @ 16,16 SAY "Enter name to display" GET qname
    READ
    qname = UPPER(TRIM(qname))
    SEEK qname
    recnumb = RECNO()
    COUNT WHILE UPPER(name) = qname TO howmany
```

والجزء السادس من البرنامج يقوم بتحذير المستخدم فى حالة عدم العثور على الاسم المطلوب . ويلاحظ هنا استخدام دالة الماكرو فى عرض الاسم أو الحرف الذي أدخله المستخدم داخل السلسلة الحرفية . كما يلاحظ استخدام الدالة (CHR(7) لتشغيل جرس التنبيه . ثم تظهر رسالة للمستخدم ليضغط على أى مفتاح للرجوع الى القائمة الرئيسية مرة أخرى . ويحتوى هذا الجزء على السطور التالية :

```
IF howmany = 0
    @ 19,15 SAY "There is no & qname"
    ? CHR (7)
    WAIT "Press any key to return to main menu"
    RETURN
ENDIF(howmany = 0)
```

والجزء السابع يقوم بعرض بيانات الاسم والفصل الدراسي الخاص بالاسماء التي تشترك فى الاسم أو الحرف المطلوب حتى يقوم المستخدم باختيار رقم السجل الخاص بالاسم المطلوب من هذه السجلات . ويلاحظ أن أمر البحث قد تم استخدامه مرة ثانية ، وذلك لأن مؤشر السجلات (Record Pointer) يكون قد تحرك نتيجة استخدام الأمر (COUNT) ويراد اعادته مرة ثانية الى أول سجل يحقق الشرط . ويتم أولاً تحريك المؤشر الى أول الملف عن طريق الأمر (GO TOP) حتى يبدأ البحث من

أول الملف . ثم يتم عرض بيانات السجلات عن طريق الأمر (DISPLAY) . ويظهر سؤال للمستخدم عن رقم السجل الذي يراد عرض بياناته في التقرير ويتم تخزين هذا الرقم في المتغير (recnumb) . ثم يتم الذهاب الى هذا السجل استعداداً لطابعته .

وفي حالة العثور على سجل واحد يحقق الشرط أى أن المتغير (howmany) تكون قيمته (1) . أى في حالة عدم تحقق الشرط (howmany>1) . في هذه الحالة يتم استخدام مرشح (Filter) للحصول على السجل الذي يحقق الشرط .

وهذا الجزء يحتوى على السطور التالية :

```
IF howmany > 1
  CLEAR
  GO TOP
  SEEK qname
  DISPLAY name, class WHILE UPPER(name) = qname
  @ ROW()+2,15 SAY "which one do you want";
  GET recnumb
  READ
  GOTO recnumb
ELSE
  SET FILTER TO UPPER(name) = qname
ENDIF(howmany>1)
ENDIF(qchoice = Q)
```

والجزء الثامن يسأل المستخدم اذا كان يريد طباعة التقرير أو الاكتفاء بعرضه على الشاشة . ويتم تخزين اجابة المستخدم في المتغير (toprint) . ويحتوى هذا الجزء على السطور التالية :

```
toprint = "N"
CLEAR
@ 10,5 SAY "Send records to printer? Y/N" GET toprint;
PICTURE "!"
READ
```

والجزء التاسع يؤدي الى توجيه التقرير الى الشاشة أو الطابعة حسب اختيار المستخدم الموجود في المتغير (toprint) . فاذا كان هذا الاختيار (Y) يتم

توجيهه الى الطابعة ، واذا كان غير ذلك يتم توجيهه الى الشاشة . ثم يتم تنفيذ البرنامج (rep.prg) عندما يكون المتغير (repchoice) محتويا على الرقم (1) . كما يتم تنفيذ البرنامج (label.prg) عندما يكون المتغير (repchoice) محتويا على الرقم (2) . وهذا الجزء يحتوى على السطور التالية :

```
IF toprint = "Y"
    SET DEVICE TO PRINT
ELSE
    SET DEVICE TO SCREEN
ENDIF
DO CASE
    CASE repchoice = 1
        DO rep
    CASE repchoice = 2
        DO label
ENDCASE
```

والجزء العاشر يتم من خلاله ايقاف التقرير على الشاشة مؤقتا (Pause) حتى يستطيع المستخدم قراءة بيانات التقرير ثم يضغط على أى مفتاح للرجوع الى قائمة التقارير مرة ثانية . ويحتوى هذا الجزء على السطور التالية :

```
IF toprint <> "Y"
    @ 24,1 CLEAR
    WAIT "Press any key to return to menu"
ENDIF
```

والجزء الحادى عشر يتم من خلاله افلاق المرشح (Filter) حتى يعود الملف الى حالته الأولى . ثم يتم الرجوع الى القائمة الرئيسية باستخدام الأمر (RETURN) .

٩ - ١ البرنامج (rep)

يتم كتابة هذا البرنامج بديلا عن استخدام قوائم برنامج المساعد (Assistant) في تصميم شكل التقرير . حيث يتم تصميم التقرير باستخدام مجموعة من الأوامر (@...SAY) داخل حلقة تكرارية يتم تكرارها حتى نهاية سجلات الملف (EOF) . وذلك لعرض التقارير كلها فى حالة طلب المستخدم ذلك . وهذا البرنامج يكون كالآتى :


```

DO WHILE .NOT. EOF()
CLEAR
    @ 2, 2 SAY "Cadet name : "
    @ 2,37 SAY NAME
    @ 3, 2 SAY "CLASS : "
    @ 3,38 SAY CLASS
    @ 4, 2 SAY "ADDRESS : "
    @ 4,38 SAY ADDRESS
    @ 5, 2 SAY "Secondary school average : "
    @ 5,38 SAY SEC_SCHOOL
    @ 6, 2 SAY "Date of entering the institute : "
    @ 6,38 SAY DT_ENT
    @ 7, 2 SAY "Telephone no : "
    @ 7,38 SAY T_NO
    @ 8, 2 SAY "Blood Type: "
    @ 8,32 SAY BLOOD
    @ 9, 2 SAY "Religion : "
    @ 9,38 SAY RELIGION
    @ 10, 2 SAY "Nationality : "
    @ 10,38 SAY NATION
    @ 11, 2 SAY "Birth Date : "
    @ 11,38 SAY B_DATE
    @ 12, 2 SAY "Birth place : "
    @ 12,38 SAY B_PLACE
    @ 13, 2 SAY "Father name: "
    @ 13,38 SAY FATH_NAME
    @ 14, 2 SAY "Mother name: "
    @ 14,38 SAY MOTH-NAME
    @ 15, 2 SAY "Father salary: "
    @ 15,38 SAY F_SALARY
    @ 16, 2 SAY "Father Job: "
    @ 16,38 SAY F_JOB
    @ 17, 2 SAY "Number of brothers : "
    @ 17,38 SAY NO_BROTHER
    @ 18, 2 SAY "Date of getting Sec-School: "
    @ 18,38 SAY DATE_SEC
    @ 19, 2 SAY "Hobbies"
    @ 19,38 SAY HOBBIES

```

```
@ 20, 2 SAY "Nearest relatives"
@ 20,38 SAY N_RELATIVE

WAIT "Press any key to continue"
CLEAR "
@ 1,40 SAY "Notes"
DISPLAY OFF NOTES

* - - - Ask the user if he want to stop displaying
KEYPRESS = 0
@ 22,12 SAY "Press any key to continue or (Q) to stop
DO WHILE KEYPRESS = 0
    KEYPRESS = INKEY()
ENDDO(KEYPRESS)
IF KEYPRESS = 113 .OR. KEYPRESS = 81
    CLEAR
    RETURN
ENDIF
SKIP
ENDDO(while not EOF)
```

ويلاحظ أن هذا البرنامج يحتوي على سطرين لكل حقل . السطر الأول يتم بواسطته عرض اسم هذا الحقل . لذلك يتم كتابة الاسم كسلسلة حرفية (string) . والسطر الثاني يتم بواسطته عرض محتويات هذا الحقل . كما يلاحظ من الاحداثيات (X,Y) التي تلي الحرف (E) أن اسم الحقل ومحتويات الحقل يتم عرضهما على سطر واحد على الشاشة . لذلك فعند عرض سجل معين على الشاشة تظهر الشاشة الموضحة بالشكل (١ - ٢) مثلا .

كما يلاحظ أيضا عرض حقل للملاحظات (Notes) في شاشة مستقلة . وذلك بمسح الشاشة السابقة عند ضغط المستخدم على أى مفتاح .

والجزء الثاني من البرنامج الذي يبدأ بإنشاء المتغير (KEYPRESS) يقوم بعرض رسالة للمستخدم لاييقاف عرض السجلات في حالة عرض السجلات كلها . وذلك عندما يريد أن يوقف عرض هذه السجلات ويعود الى قائمة التقارير مرة ثانية .

ويلاحظ استخدام الدالة (INKEY()) في اختبار الشفرة الخاصة بالمفتاح الذي يضغط عليه المستخدم . فإذا كانت هذه الشفرة تساوي (81) أو (113) فإن هذا

يعني أن المستخدم قد ضغط على مفتاح (Q) أو مفتاح (q) على الترتيب . وفي هذه الحالة يتم الخروج من البرنامج والمودة الى قائمة التقارير . أما إذا ضغط على أى مفتاح آخر ، فإن هذا يؤدي الى عرض بيانات السجل التالي .

CADET NO	:	8960
NAME	:	MOHAMED ALY SALEM
CLASS	:	55P
DATE OF ENTERING AIR ACADEMY	:	09/01/85
BLOOD CLASS	:	A
ADDRESS	:	13 - ABBAS ELAKKAD
TELEPHONE NO	:	2603555
RELIGION	:	MUSLIM
NATIONALITY	:	EGYPTIAN
HOBBIES	:	FOOTBALL
BIRTH DATE	:	01/22/67
BIRTH PLACE	:	TANTÀ
SECONDARY SCHOOL AVERAGE	:	70.00
DATE OF GETTING SECONDARY SCHOOL	:	1985
FATHER NAME	:	ALY SALEM
MOTHER NAME	:	FATMA MAHMOUD
FATHER SALARY	:	800
NUMBER OF BROTHERS	:	5

شكل (٩ - ٢)

٩ - ٢ البرنامج (label)

يتم كتابة هذا البرنامج لتصميم التقارير المختصرة بدلا من استخدام برنامج المساعد (Assistant) . وهو يماثل البرنامج (rep) السابق شرحه ولذلك سوف نكتفي بكتابة سطور البرنامج دون شرحها وهي كالآتي :

```
* - - - - - label.prg
* - - - - - Program for making labels
R = 2
GO TOP
DO WHILE .NOT. EOF()
```

```
@ R,2 SAY CADET_NO
@ R+1,2 SAY NAME
@ R+2,2 SAY GRL_SECT
@ R+3,2 SAY ADDRESS
@ 17,2 SAY "Press any key to continue or + ;
"(Q) to stop"
KEYPRESS = 0
DO WHILE KEYPRESS = 0
    KEYPRESS = 0
ENDDO(KEYPRESS)
IF KEYPRESS = 113 .OK. KEYPRESS = 81
    RETURN
ENDIF
CLEAR
SKIP
?
ENDDO
RETURN
```

الفصل العاشر

برنامج التصحيح

يساعد هذا البرنامج على وصول المستخدم الى السجل المطلوب وتصحيحه .
وكالعادة يتم أولا كتابة الخطوات الأولية (PSEUDOCODE) كالآتي :

- ١ - يتم تكوين حلقة تكرارية لتعديل السجلات .
- ٢ - يتم السؤال عن الاسم أو الحرف المطلوب البحث عنه .
- ٣ - في حالة عدم ادخال أى اسم ، يتم الرجوع الى القائمة الرئيسية .
- ٤ - يتم تحويل الاسم الى حروف كبيرة حتى يماثل ملف الفهرس .
- ٥ - يتم البحث عن الاسم المطلوب .
- ٦ - يتم حساب عدد السجلات التي تحتوى على نفس الاسم .
- ٧ - في حالة عدم العثور على أى سجل يحتوى على هذا الاسم يتم تحذير المستخدم حتى يدخل اسما آخر .
- ٨ - في حالة وجود عدة سجلات تحتوى على نفس الاسم يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل .
- ٩ - عند الوصول الى السجل المطلوب يتم عرض بياناته على شاشة الادخال التي سبق تصميمها .
- ١٠ - يتم العودة الى القائمة الرئيسية بعد تعديل الحقول المطلوبة .

ثم يتم كتابة البرنامج كالآتي :

```
***** Cadedit.prg
* Lookup and edit data in the cadets database

* - - - - - Set up a loop for editing records
entering = .T.
DO WHILE entering
    SET EXACT OFF
    GO TOP
    * - - - - - Ask for name of person to lookup
    CLEAR
    lookup = SPACE(4)
    @ 10,12 SAY " Enter name of person to edit"
    @ 12,12 SAY "or just press Return to exit" ;
    GET lookup
    READ

    * - - - - - If no name entered , skip all
    * - - - - - commands between here and enddo
    IF lookup = " "
        entering = .F.
```

```

      LOOP
    ENDIF

```

```

* - - - - - convert lookup to uppercase to
* - - - - - match index file, and trim it
lookup = UPPER (TRIM(lookup))

```

```

SEEK lookup

```

```

mrecord = RECNO()

```

```

* - - - - - count how many there are
COUNT WHILE UPPER (NAME) = lookup TO howmany
IF howmany = 0

```

```

    @ 20,10 SAY "There is no & lookup"
    @ 22,10 SAY "press any key to try again"
    ? CHR(7)

```

```

    WAIT

```

```

    mrecord = 0

```

```

ENDIF(howmany = 0)

```

```

* - - - - - if more than one record has that
* - - - - - name, get more information

```

```

IF howmany > 1

```

```

    CLEAR

```

```

    mrecord = 0

```

```

    SEEK lookup

```

```

    DISPLAY NAME, CLASS WHILE UPPER(NAME) = lookup

```

```

    @ ROW()+3,10 SAY "Edit which record # ?" GET ;

```

```

    mrecord PICTURE "9999"

```

```

    READ

```

```

ENDIF

```

```

* - - - - - If there is a record number greater
* - - - - - than zero, edit the record

```

```

IF mrecord > 0

```

```

    CLEAR

```

```

    GOTO mrecord

```

```

    SET FORMAT TO cadets

```

```

    READ

```

```

    CLOSE FORMAT

```

```

ENDIF

```



```
ENDDO (WHILE entering)
RETURN
```

وببدأ هذا البرنامج باسم البرنامج ووظيفته كالمعتاد . ثم يتم تكوين حلقة تكرارية تعتمد على المتغير المنطقي (entering) . وحيث أن قيمته في البداية تكون (.T.) أى صحيح ، لذلك يتم تنفيذ الحلقة التكرارية أول مرة . ثم يتم التحكم من داخل الحلقة فى قيمة المتغير المنطقي (entering) لاستمرار تنفيذ الحلقة أو إيقاف تنفيذها .

وهذا الجزء يتكون من السطور التالية :

```
entering = .T.
DO WHILE entering
```

وفى الجزء الثانى من البرنامج يبدأ تنفيذ الحلقة التكرارية . ويتم انشاء المتغير (lookup) الذي يكون طوله (4) حروف . وذلك حتى يدخل المستخدم فيه الحروف الأولى من الاسم المطلوب البحث عنه . ثم يتم عرض رسالة للمستخدم لادخال الحروف المطلوبة وتخزين هذه الحروف فى المتغير (lookup) . ويحتوى هذا الجزء على السطور التالية :

```
CLEAR
lookup = SPACE(4)
@ 10,12 SAY " Enter name of person to edit"
@ 12,12 SAY "or just press Return to exit" ;
GET lookup
READ
```

وفى الجزء الثالث يتم تخزين القيمة (.F.) فى المتغير (entering) فى حالة ضغط المستخدم على مفتاح الادخال دون كتابة أى حروف . وفى هذه الحالة يتم الانتقال الى أول الحلقة التكرارية بواسطة الأمر (LOOP) . وحيث أن المتغير (entering) يكون غير صحيح (False) ، فلا يتم تنفيذ الحلقة التكرارية وتعود القائمة الرئيسية للظهور . ويحتوى هذا الجزء على السطور التالية :

```
IF lookup = " "
entering = .F.
LOOP
ENDIF
```

وفي الجزء الرابع من البرنامج يتم البحث عن الحروف التي يدخلها المستخدم خلال حقل الاسم (name) . مع ملاحظة أنه سبق فتح ملف الفهرس من خلال البرنامج الرئيسي . كما أن هذا الفهرس قد تم انشاؤه بحيث يتضمن الأسماء بحروف كبيرة (Upercase) حتى يسهل مطابقتها على الاسم الذي يدخله المستخدم . كما يتم تحويل الحروف التي يدخلها المستخدم أيضا إلى حروف كبيرة . لذلك فإن ادخال المستخدم للحروف الكبيرة أو الصغيرة لا يؤثر في البحث عن هذه الحروف . كما أن استخدام الدالة (TRIM) يؤدي إلى إلغاء أي مسافات موجودة بعد الحروف التي يدخلها المستخدم . وهذا يتيح له ادخال حرف واحد مثلا دون الخوف من تأثير المسافات الثلاثة الباقية على البحث .

وهذا الجزء يحتوي على السطور التالية :

```
lookup = UPPER (TRIM(lookup))
SEEK lookup
mrecord = RECNO()
```

ويجب ملاحظة أن الأمر (SEEK) وظيفته توجيه مؤشر السجلات (Record Pointer) إلى السجل المطلوب . كما يتم عن طريق السطر التالي له تخزين رقم هذا السجل في متغير الذاكرة (mrecord) . وذلك لكي يسهل الذهاب إلى هذا السجل بعد ذلك .

وفي الجزء الخامس من البرنامج يتم حساب عدد السجلات التي تحقق شرط الاسم وتخزين هذا العدد في المتغير (howmany) . بحيث أن الملف مفهرس على حقل الاسم (name) ، لذلك يفضل استخدام (WHILE) في البحث بدلا من (FOR) حيث أنه يقوم بتجميع السجلات التي تحقق الشرط . ولذلك يكون البحث أسرع بواسطة (WHILE) .

بعد ذلك يقوم البرنامج باختبار العدد الذي تم تخزينه في المتغير (howmany) . فإذا كان هذا العدد صفرا فإن هذا يعني أنه لا يوجد سجل يبدأ بالحروف التي أدخلها المستخدم . ولذلك تظهر الرسالة البيئة ، والتي يتم خلالها كتابة الحروف التي أدخلها المستخدم في المتغير (lookup) باستخدام الماكرو (&) . كما يتم تحذير المستخدم عن طريق الدالة (CHR(7)) التي تؤدي إلى تشغيل الجرس . وهذا الجزء يحتوي على السطور التالية :

```
IF howmany = 0
  @ 20,10 SAY "There is no & lookup"
  @ 22,10 SAY "press any key to try again"
  ? CHR(7)
  WAIT
```

```
mrecord = 0
ENDIF(howmany = 0)
```

وفي الجزء السادس يتم دراسة حالة أخرى وهي وجود أكثر من سجل يحقق الشرط ، أى أن (howmany) أكبر من (١) . في هذه الحالة يقوم البرنامج بمرض بعض بيانات هذه السجلات حتى يستطيع المستخدم تمييز السجل الذى يريد تعديله . كما يتم عرض أرقام السجلات الخاصة بهذه السجلات . ويتم سؤال المستخدم عن السجل المطلوب ليقوم بادخال رقم هذا السجل . وهذا الجزء يحتوى على السطور التالية :

```
IF howmany > 1
  CLEAR
  mrecord = 0
  SEEK lookup
  DISPLAY NAME , CLASS WHILE UPPER(NAME) = lookup
  @ ROW()+3,10 SAY "Edit which record # ?" ;
  GET mrecord PICTURE "9999"
  READ
ENDIF
```

وفي الجزء السابع يتم اختيار المتغير (mrecord) فإذا كان أكبر من صفر، فإن هذا يعنى أنه تم إيجاد أحد السجلات المطابقة سواء من خلال الشرط الأول (howmany = 0) أو من خلال الشرط الثانى (howmany > 1) أو من خلال الحالة الوحيدة المتبقية وهي (howmany = 1) . لذلك يتم الذهاب الى هذا السجل باستخدام الأمر (GOTO) . ثم يتم فتح ملف التشكيل (Format file) الذى يؤدى الى عرض شاشة الادخال الخاصة بملف قاعدة البيانات المفتوح . وهذا يتيح للمستخدم ادخال التعديلات المطلوبة ، ثم يتم اغلاق ملف التشكيل مرة ثانية . وهذا الجزء يحتوى على السطور التالية :

```
IF mrecord > 0
  CLEAR
  GOTO mrecord
  SET FORMAT TO cadets
  READ
  CLOSE FORMAT
ENDIF
ENDDO (WHILE entering)
RETURN
```


الفصل الحادي عشر

برنامج مسح المجالات

يساعد هذا البرنامج على وصول المستخدم الى سجل أو عدة سجلات ومسحها .
والخطوات الأولية (PSEUDOCODE) لهذا البرنامج تكون كالآتي :

- ١ - يتم تكوين حلقة تكرارية لمسح السجلات .
 - ٢ - يتم السؤال عن الاسم المطلوب مسح السجل الخاص به .
 - ٣ - عند عدم ادخال أى اسم ، يتم الرجوع الى القائمة الرئيسية .
 - ٤ - يتم تحويل الاسم الى حروف كبيرة (Uppercase) .
 - ٥ - يتم البحث عن الاسم المطلوب .
 - ٦ - يتم حصر عدد السجلات التي تحتوى على نفس الاسم .
 - ٧ - فى حالة عدم العثور على أى سجل يحتوى على هذا الاسم ، يتم تحذير المستخدم حتى يدخل اسما آخر .
 - ٨ - فى حالة وجود عدة سجلات تحتوى على نفس الاسم ، يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل .
 - ٩ - عند الوصول الى السجل المطلوب ، يتم سؤال المستخدم مرة ثانية للتأكد من رغبته فى مسح هذا السجل .
 - ١٠ - عند تأكد المستخدم من رغبته فى مسح السجل يتم وضع علامة على هذا السجل تمهيدا لمسحه .
 - ١١ - يتم السماح للمستخدم بادخال أسماء أخرى حتى ينتهى من تحديد الأسماء التي يريد مسحها .
 - ١٢ - يتم حصر عدد السجلات التي تم وضع علامات عليها لمسحها .
 - ١٣ - طالما كانت هناك سجلات عليها علامات المسح (Marked for deletion) ، يتم تنفيذ الآتي :
 - يتم عرض بيانات السجلات التي تم وضع علامات عليها .
 - يتم التأكد من رغبة المستخدم فى مسح هذه السجلات كلها مسحا دائما (Permanently) .
 - اذا أراد المستخدم استرجاع بعض هذه السجلات يتم اعطاؤه الفرصة لاستعادة أحد هذه السجلات .
 - عندما يتأكد المستخدم من رغبته فى مسح السجلات الباقية يتم مسح هذه السجلات مسحا دائما (Permanently) .
 - ١٤ - تعود القائمة الخاصة بالمسح للظهور حتى يمكن تكرار العملية مع سجلات أخرى .
 - ١٥ - بعد الانتهاء يتم الرجوع الى القائمة الرئيسية .
- بعد كتابة الخطوات الأولية (PSEUDOCODE) ، يتم كتابة البرنامج الذى يقوم بتنفيذها كالآتي :

```
* - - - - - Set up loop for deleting records .
entering = .T.
DO WHILE entering
  * - - - - - Ask for name of person to lookup
  CLEAR
  lookup = SPACE(4)
  @ 10,12 SAY "Enter name of person to delete"
  @ 12,12 SAY "or just press return to exit"
  GET lookup
  READ
  * - - - - - If no name entered , skip all
  * - - - - - commands between here and Enddo
  IF lookup = " "
    entering = .F.
    LOOP
  ENDIF (lookup = " ")

  * - - - - - convert lookup to uppercase to match
  * - - - - - index file
  lookup = UPPER(lookup)

  * - - - - - Try to find requested name, and
  * - - - - - remember record number
  SEEK lookup
  mrecord = RECNO()

  * - - - - - Count howmany there are
  COUNT WHILE UPPER (NAME) = lookup TO howmany
  * - - - - - If no record has that name , warn the user
  * - - - - - to try again
  IF howmany = 0
    @ 20,10 SAY "There is no & lookup"
    @ 22,10 SAY "press a key to try again"
    ? CHR(7)
    WAIT " "
    mrecord = 0
  ENDIF (howmany = 0)
```



```

* - - - - - If more than one record has that name
* - - - - - display records to the user.
IF homany > 1
    CLEAR
    mrecord = 0
    SEEK lookup
    LIST NAME , CLASS WHILE UPPER(NAME) = lookup ;
    @ ROW()+3,10 SAY "Delete which one?"
    GET mrecord PICTURE "9999"
    HEAD
ENDIF

* - - - - -If the value of (mrecord) greater than
* - - - - - zero double check then delete
IF mrecord > 0
    GOTO mrecord
    CLEAR
    DISPLAY NAME , CLASS
    ?
    WAIT "Delete this record? (Y/N)" TO answer

    * - If answer is yes, mark record for deletion
    IF UPPER(answer) = "Y"
        DELETE RECORD mrecord
        ENDIF(answer)
    ENDIF(mrecord > 0)
    ENDDO (while entering)

    * - Before exiting , verify deletion and pack
    COUNT FOR DELETED() TO nodels
    oktopack = "N"
    DO WHILE oktopack = "N" .AND. nodels > 0
        CLEAR
        ? "Records to be deleted .."
        ?
        DISPLAY NAME , CLASS FOR DELETED()
        @ 23,1 SAY "Delete all these? (Y/N)" ;

```

```

GET oktopack PICTURE "!"
READ
IF oktopack < > "Y"

* - if not ok to pack , recall a record
delrec = 0
@ 23,1 SAY "Recall which one";
GET delrec PICTURE "9999"
READ
* - if record number entered and record
* - is indeed deleted, recall it
IF delrec > 0
GOTO delrec
IF DELETED()
RECALL RECORD delrec
models = models -1
ENDIF
ENDIF
ELSE

* - - - - if ok to pack , pack and show
* - - - - the process of packing
SET TALK ON
PACK
SET TALK OFF
ENDIF (oktopack)
ENDDO(oktopack)
RETURN

```

الهدف من هذا البرنامج هو البحث عن سجل معين ومسحه . والجزء الخاص بالبحث لا يختلف عن برنامج التصحيح أو برنامج التقارير . حيث يقوم البرنامج بمرض رسالة للمستخدم لادخال الاسم المطلوب البحث عنه . ثم البحث عن هذا الاسم باستخدام أوامر البحث المعروفة ثم عرض بيانات السجلات التي تشترك في هذا الاسم حتى يقوم المستخدم باختيار سجل محدد منها عن طريق رقم السجل .

ولكن المطلوب من البرنامج بعد ذلك مسح هذا السجل . وعملية المسح تتم بشيء من الخطورة حيث أن المستخدم قد يسمح سجلا ثم يكتشف بعد ذلك أنه قد

مسح سجل مطلوب من طريق الخطأ . ولذلك فإن البرنامج يجب أن يتيح للمستخدم عدة مراحل من الاختبار والتحذير حتى يتأكد تماما أن هذا السجل هو السجل المطلوب مسحه .

والجزء الأول من البرنامج لايحتاج الى اعادة شرحه حيث أنه سبق شرحه في برنامج التصحيح وبرنامج التقارير .

والجزء الجديد هو الذى يبدأ بعد الوصول الى رقم السجل المطلوب مسحه . وهو الجزء الذى يبدأ بمجموعة السطور التالية :

```
* - - - - If the value of (mrecord) greater than zero,
* - - - - double check then delete
IF mrecord > 0
    GOTO mrecord
    CLEAR
    DISPLAY NAME , CLASS
    ?
    WAIT "Delete this record? (Y/N)" TO answer

    * - - If answer is yes, mark record for deletion
    IF UPPER(answer) = "Y"
        DELETE RECORD mrecord
    ENDIF(answer)
ENDIF(mrecord > 0)
ENDDO (while entering)
```

ويمكن للمستخدم من خلال الحلقة التكرارية الأولى (WHILE entering) تحديد عدة سجلات مطلوب مسحها . وفي كل مرة يتم عرض بيانات السجل عليه حتى يتأكد أن هذا هو السجل المطلوب . وهذه العملية تؤدي في النهاية الى وضع علامات أمام عدة سجلات تمهيدا لمسحها .

وبعد ذلك يتم تكوين حلقة تكرارية أخرى تتيح للمستخدم المسح النهائي لهذه السجلات . وهذه الحلقة التكرارية تظهر في السطور التالية :

```
* - - - - Before exiting , verify deletion and pack
COUNT FOR DELETED() TO nodels
oktopack = "N"
```

```

DO WHILE oktopack = "N" .AND. nodels > 0
  CLEAR
  ? "Records to be deleted .."
  ?
  DISPLAY NAME , CLASS FOR DELETED()
  @ 23,1 SAY "Delete all these? (Y/N)";
  GET oktopack PICTURE "!"
  READ
  IF oktopack < > "Y"

    * - - - - - if not ok to pack , recall a record
    delrec = 0
    @ 23,1 SAY "Recall which one (by record#)";
    GET delrec PICTURE "9999"
    READ

    * - - - - -if record number entered and record is
    * - - - - - indeed deleted, recall it
    IF Delrec > 0
      GOTO delrec
      IF DELETED()
        RECALL RECORD delrec
        nodels = nodels -1
      ENDIF
    ENDIF
  ELSE

    * - - - - - if ok to pack , pack and show
    * - - - - - the process of packing
    SET TALK ON
    PACK
    SET TALK OFF
  ENDIF (oktopack)
ENDDO(oktopack)
RETURN

```

وهذه المجموعة من السطور تبدأ بحساب عدد السجلات التي تم وضع علامات عليها

باستخدام الأمر (COUNT) وتخزين هذا العدد في المتغير (models). ويجب ملاحظة أن الأمر (COUNT) يقوم بإنشاء المتغير ألبا، أي ليست هناك حاجة لإنشاء المتغير (models) قبل استخدامه. ثم تبدأ الحلقة التكرارية ويلاحظ قبلها إنشاء المتغير (oktopack) وإعطاؤه القيمة ("N") وذلك حتى يبدأ تنفيذ الحلقة التكرارية مرة واحدة بصرف النظر عن القيمة التي يدخلها المستخدم بعد ذلك في هذا المتغير. ويتوقف تنفيذ الحلقة التكرارية على شرطين. الأول هو (oktopack = "N") وهذا يكون صحيحا في البداية. والشرط الثاني هو (models > 0) وهذا يعني أن هناك سجلات تم وضع علامات عليها. أما إذا لم تكن هناك سجلات من هذا النوع فهذا معناه أن المستخدم لم يدخل أي اسم في البداية لمسحه، أو أن الاسم الذي تم ادخاله لم يتم العثور عليه. وفي هذه الحالة لا يتم تنفيذ الحلقة التكرارية ويعود البرنامج إلى القائمة الرئيسية.

وعندما يتحقق الشرطان تبدأ الحلقة التكرارية في التنفيذ وتظهر بيانات الاسم والفرقة (CLASS) الخاصة بالسجلات التي تم وضع علامات عليها. كما يظهر سؤال للمستخدم إذا كان يريد مسح هذه السجلات كلها أم يريد استعادة أحدها. فإذا أراد استعادة أحدها فإن الشرط الموجود مع الأمر ("Y" < oktopack) يتحقق. وهذا يؤدي إلى ظهور سؤال آخر للمستخدم عن رقم السجل الذي يريد استعادته، مع ملاحظة أن السجلات تكون معروضة على الشاشة ومعها أرقام السجلات. وعندما يدخل المستخدم رقم السجل المطلوب استعادته، يتم تخزين هذا الرقم في المتغير (delrec). ونتيجة لذلك فإن الشرط (IF delrec > 0) يتحقق وينتقل المؤشر إلى هذا السجل. ثم يتم اختبار هذا السجل عن طريق الدالة (DELETED()) للتأكد أن السجل تم وضع علامة عليه لمسحه. وفي هذه الحالة يتم استعادة هذا السجل باستخدام الأمر (RECALL) ثم يتم انقاص عدد السجلات الموجودة في المتغير (models) بمقدار (1). وتكرر هذه العملية حتى يقوم المستخدم باستعادة جميع السجلات التي لا يريد مسحها. فإذا كان هناك سجلات باقية يراد مسحها، فإن المستخدم يجب على السؤال الخاص بمسح جميع السجلات بكتابة (Y) وهذا يؤدي إلى عسدم تحقق الشرط ("Y" < oktopack) وبالتالي تنفيذ الأوامر بعد (ELSE). ومن خلال هذه الأوامر يتم اتمام عملية المسح باستخدام الأمر (PACK). ويلاحظ أنه تم استخدام الأمر (SET TALK ON) قبل استخدام الأمر (PACK) وذلك حتى يرى المستخدم الرسالة الدالة على اتمام عملية المسح. ثم يتم الرجوع إلى البرنامج الرئيسي باستخدام الأمر (RETURN).

3

الجزء الثالث

نظام المخازن

- * توصيف النظام
- * برنامج القائمة الرئيسية
- * برنامج تشغيل الملف الرئيسى
- * برنامج تشغيل ملف المبيعات
- * برنامج تشغيل ملف الاضافة
- * برنامج تحديث البيانات

الفصل الثاني عشر

توصيف النظام

من النظم الشائعة الاستخدام في الوقت الحالي نظم التحكم في المخزون (Stock Control) وهي النظم التي تتيح التحكم في كمية البضائع المخزونة ومتابعة البضائع الواردة والبضائع المصروفة . واستخدام قواعد البيانات في هذه النظم أدى الى كفاءة ملحوظة في السيطرة على هذه العملية . وقد روعي في هذا الكتاب شرح برنامج مخازن متكامل لأنه يوفر لمخطط البرامج شرحا لمهارات متعددة في كتابة البرامج حيث أنه يستخدم نظام التعديل المجمع (Batch Updating) من خلال ملفات الحركة (Transaction Files) . كما يمكن ربط هذا البرنامج ببرنامج حسابات العملاء (Accounts Receivable) المشروح في الجزء الرابع لانشاء نظام متكامل .

١٢ - ١ تصميم النظام

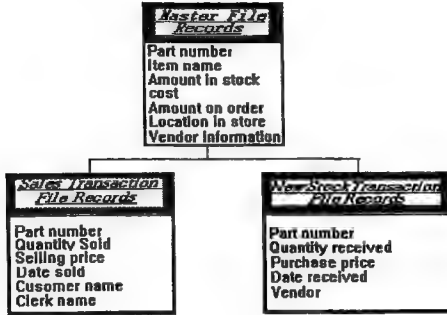
يتطلب نظام المخازن بصفة عامة تصميم عدة ملفات للبيانات أولها يسمى الملف الرئيسي (Master File) . وهذا الملف الرئيسي يشتمل على بيانات عن الأصناف الموجودة حاليا بالمخزن مثل كمية هذه الأصناف بالإضافة الى سعر كل صنف والكمية تحت الطلب من هذا الصنف (In Order) وحسب الطلب له (Reorder Point) واسم البائع (Vendor) الذي يتم شراء هذا الصنف منه وعنوانه . كما يمكن أيضا أن يتضمن بيانات عن مكان كل صنف في المخزن حتى يسهل على المستخدم الوصول اليه .

ويتطلب نظام المخازن أيضا تصميم ملفات تتضمن بيانات عن حركة الأصناف الموجودة في المخزن ، وتسمى ملفات الحركة (Transaction Files) . وأحد هذه الملفات يشمل بيانات عن حركة بيع الأصناف مثل بيانات العميل ، ومتى تم البيع وبأي كمية ورقم فاتورة الشراء ويسمى هذا الملف ملف المبيعات (Sales File) .

وهناك ملف حركة آخر يتضمن بيانات عن الأصناف الجديدة التي يتم ادخالها الى المخزن (Newstock File) .

ويمكن ملاحظة العلاقة بين هذه الملفات الثلاثة من الشكل (١٢ - ١)

ويلاحظ من الشكل أن هناك حقلا مشتركا بين الملفات الثلاثة وهو حقل رقم الجزء (Part number) . وهذا الحقل يسمى حقل المفتاح (Key Field) الذي يتم عن طريقه ربط الملفات الثلاثة .



شكل (١٢ - ١)

١٢ - ٢ حقل المفتاح (Key Field)

كما سبق الايضاح في الجزء الأول من الكتاب ، فعند تصميم نظام يتعامل مع عدة ملفات قوائم بيانات ، يجب تحديد حقل مشترك بين هذه الملفات حتى يمكن ربط هذه الملفات من خلاله . وهذا الحقل يجب أن يكون منفردا (Unique) لكل سجل في الملف الرئيسي (Master File) . كما يجب أن يكون بنفس الاسم والنوع والطول في الملفات الثلاثة .

وفي معظم نظم المعلومات يكون هناك رقم أو كود يمكن استخدامه كحقل مفتاح لأنه يكون منفردا (Unique) . ففي البنوك مثلا يستخدم رقم الحساب كحقل مفتاح حيث لا يوجد عميلان يشتركان في رقم حساب واحد . وكذلك على مستوى الدولة يستخدم الرقم الشخصي (Social Security Number) كمفتاح للحصول

على بيانات أى شخص . وبالنسبة لنظام المخازن ، فإن أنسب رقم يمكن استخدامه كحقل مفتاح هو رقم الجزء . حيث أن كل جزء له رقم جزء (Part Number) منفرد (Unique) .

١٢ - ٣ وظائف النظام

يجب تصميم نظام يسمح للمستخدم بمتابعة البضائع المخزونة والبضائع تحت الطلب ويمكن هذه البضائع فى المخزن . وعند ادخال أى أصناف غير موجودة أصلا بالمخزن يتم تحديد رقم الجزء (Part Number) الخاص بها وإضافة كمية كل صنف إلى الملف الرئيسى . كما يجب أن يوفر النظام التقارير التى توضح حالة الأصناف بالمخزن وتحدد الأصناف التى يجب طلبها . كما يقوم بإنشاء طلبات الشراء آليا .

كما يجب أن يسمح النظام للمستخدم بمتابعة حركة الأصناف ، من حيث الأصناف التى يتم بيعها والشخص الذى يقوم بعملية البيع والعميل الذى تباع له الأصناف ، وتاريخ البيع ورقم الفاتورة . كما يجب أن يسمح أيضا بمتابعة الأصناف الواردة لتعويض الأصناف الناقصة فى المخزن . كما يجب أن يقوم النظام آليا بتحديث البيانات الموجودة فى الملف الرئيسى (Master File) من ملف المبيعات (Sales) وملف الوارد (New Stock) .

١٢ - ٤ تحديد حقول الملفات

فى معظم الأحيان تكون أسهل وسيلة لتحديد حقول ملف قاعدة البيانات هى تحديد المخرجات المطلوبة من هذا الملف وهى التقارير المطلوب انشاؤها . فمثلا لتحديد حقول الملف الرئيسى (Master File) يتم دراسة محتويات التقارير المطلوبة من هذا الملف . فأحد هذه التقارير هو التقرير الذى يوضح موقف أو حالة الأصناف الموجودة فى المخزن . هذا التقرير يشتمل على رقم الجزء واسم الجزء والكمية الموجودة من هذا الجزء فى المخزن وسعر الشراء ويمكن هذا الجزء فى المخزن والكمية تحت الطلب من هذا الجزء .

وهناك تقرير آخر يسمى تقرير إعادة الطلب (Reorder Report) يتم من خلاله عرض قائمة بالأصناف المطلوبة . كما أن هناك تقريراً آخر يسمى تقرير تحت الطلب (On Order Report) ، ويعرض قائمة بالأصناف الجارى طلبها . كما أن هناك طلبات الشراء التى يقوم المستخدم بإصدارها لشراء أصناف جديدة .

ولتوفير هذه المخرجات يجب أن يشتمل الملف الرئيسى (Master File) على البيانات التالية :

Part number
Item name
Quantity in stock
Purchase price
Reorder point
Quantity on order
Location in warehouse
Vendor name
Vendor address
Date of last update
Date of last order
Quantity to order

كما يجب أن يوفر النظام أيضا تقارير توضح كل حركة بيع . وهذه التقارير تحتوي على رقم الجزء والكمية ورقم الفاتورة (Invoice Number) واسم الشخص القائم بالبيع واسم العميل وسعر البيع وتاريخ البيع . ولذلك فإن ملف المبيعات (Sales File) يجب أن يحتوي على البيانات الآتية :

Part number
Invoice number
Salesperson's name
Quantity sold
Selling price
Date sold
Whether posted or not

كما أن النظام يجب أن يوفر تقارير توضح حركة الأصناف الواردة وتعرض رقم الجزء والكمية الواردة وثمان الشراء وتاريخ الورد واسم البائع (Vendor) . لذلك فإن ملف الأصناف الواردة (New Stock File) يجب أن يحتوي على البيانات الآتية :

Part number
Quantity
Purchase price
Date received
Vendor name
Whether posted or not

١٢ - ٥ تصميم قاعدة البيانات

كما سبق الايضاح فان نظام المخازن في العادة يحتوى على ثلاثة ملفات قوائم بيانات . وهي الملف الرئيسى وليكن اسمه (Master.dbf) وملفان حركة أحدهما خاص بالمبيعات ويسمى (Sales.dbf) والآخر خاص بالأصناف الواردة ويسمى (Newstock.dbf) .

١٢ - ٥ - ١ انشاء الملف الرئيسى (Master File)

يتم انشاء الملف الرئيسى عن طريق قوائم برنامج المساعد (Assistant) كما سبق الايضاح فى الكتاب الأول . كما يمكن انشاؤه أيضا باستخدام الأمر (CREATE) وذلك كالآتى :

CREATE MASTER

ولى هذه الحالة تظهر شاشة هيكل ملف قاعدة البيانات (Structure) التى يتم عن طريقها كتابة اسم كل حقل ونوعه وطوله وعدد الأرقام العشرية أن وجدت . ويتم تكوين الملف كالآتى :

	FIELD NAME	TYPE	WIDTH	DEC
1	PART_NO	Character	5	
2	P_NAME	Character	20	
3	QTY	Numeric	7	2
4	COST	Numeric	9	2
5	REORDER	Numeric	7	2
6	ON_ORDER	Numeric	7	2
7	LOCATION	Character	5	
8	VENDOR	Character	25	
9	VENDOR_ADD	Character	25	
10	DATE	Date	8	
11	ORDER_DATE	Date	8	
12	NEW_ORDER	Numeric	7	2

ومن هذا الشكل يلاحظ أن الحقل رقم (١) هو الحقل الخاص برقم الجزء (Part Number) وهو حقل حرفي يتكون من خمسة حروف .
والحقل رقم (٢) هو حقل اسم الجزء (Part Name) وهو حقل حرفي يتكون من عشرين حرفا . والحقل رقم (٣) هو حقل الكمية (Quantity) وهو حقل عددي يتكون من سبعة أرقام متضمنا رقمين عشريين . والحقل رقم (٤) هو حقل سعر الشراء (Cost) وهو حقل عددي يتكون من تسعة أرقام متضمنا رقمين عشريين . والحقل رقم (٥) هو حقل حد الطلب (Reorder Point) ويمثل الحد الذي يجب عنده إعادة طلب هذا الصنف ويتكون من سبعة أرقام متضمنا رقمين عشريين . والحقل رقم (٦) هو حقل الكمية الجارية طلبها ، وهو حقل عددي مكون من سبعة أرقام متضمنا رقمين عشريين . والحقل رقم (٧) هو حقل مكان الصنف في المخزن (Location) وهو حقل حرفي مكون من خمسة حروف . والحقل رقم (٨) هو حقل البائع (Vendor) وهو حقل حرفي مكون من خمسة وعشرين حرفا . والحقل رقم (٩) هو حقل عنوان البائع وهو حقل حرفي مكون من خمسة وعشرين حرفا . والحقل رقم (١٠) هو حقل تاريخ آخر تحديث للملف وهو حقل تاريخي مكون من ثمانية حروف . والحقل رقم (١١) هو حقل تاريخ آخر طلب للصنف (Date of last order) . والحقل رقم (١٢) هو حقل الكمية التي يجب طلبها من هذا الصنف ، وهو حقل عددي مكون من سبعة أرقام متضمنة رقمين عشريين .

ولإنشاء ملف الفهرس الخاص بهذا الملف على أن يكون الحقل الخاص برقم الجزء هو الحقل الفهرسي (Key Field) يتم كتابة السطرين التاليين من مشيرة النقطة .

```
USE Master
INDEX ON Part_no TO Master
```

١٢ - ٥ - ٢ إنشاء ملف المبيعات (Sales File)

يتم إنشاء ملف المبيعات (Sales.dbf) بنفس الطريقة كما سبق إنشاء الملف الرئيسي . ويجب ملاحظة أن حقل رقم الجزء (Part_no) يجب أن يكون موجودا في هذا الملف وبطول خمسة حروف مثل الملف الرئيسي تماما . ويتم تكوين الملف كالآتي :

توصيف النظام

FIELD NAME	TYPE	WIDTH	DEC
1 PART_NO	Character	5	
2 INVOICE_NO	Numeric	6	0
3 CLERK	Character	12	
4 CUSTOMER	Character	12	
5 QTY	Numeric	7	2
6 PRICE	Numeric	9	2
7 DATE	Date	8	
8 POSTED	Logical	1	

والحقل رقم (١) هو حقل رقم الجزء (Part number) . والحقل رقم (٢) هو حقل رقم الفاتورة (Invoice number) وهو حقل عددي مكون من ستة أرقام . والحقل رقم (٣) هو حقل اسم الموظف القائم بالبيع وهو قد يكون اسم الموظف أو كود معين خاص به حسب الحاجة . وهو حقل حرفي مكون من (١٢) حرفا . والحقل رقم (٤) هو حقل اسم العميل (Customer) وهو حقل حرفي مكون من (١٢) حرفا . والحقل رقم (٥) هو حقل الكمية المباعة وهو حقل عددي مكون من سبعة أرقام متضمنة رقمين عشريين . والحقل رقم (٦) هو حقل سعر البيع وهو حقل عددي مكون من تسعة أرقام ورقميين عشريين . والحقل رقم (٧) هو حقل تاريخ البيع ، وهو حقل تاريخي مكون من ثمانية حروف . والحقل رقم (٨) هو حقل الترحيل (Posting) وهو حقل منطقي (Logical) يستخدم للتحكم في ترحيل بيانات الصنف الى الملف الرئيسي ، وضمان عدم تكرار ترحيلها .

ولانشاء ملف الفهرس الخاص بهذا الملف يتم كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) :

```
USE Sales
INDEX ON Part_no TO Sales
```

١٢ - ٥ - ٢ انشاء ملف الاصناف الواردة

يتم انشاء ملف الاصناف الواردة (NewStock.dbf) بنفس الطريقة كما سبق الايضاح . مع ملاحظة ضرورة وجود حقل رقم الجزء (Part Number) بنفس الاسم والطول والنوع . ويتم تكوين الملف كالآتي :

FIELD NAME	TYPE	WIDTH	DEC
1 PART_NO	Character	5	
2 QTY	Numeric	7	2
3 COST	Numeric	9	2
4 DATE	Date	8	
5 VENDOR	Character	50	
6 POSTED	Logical	1	

والحقل رقم (١) هو حقل رقم الجزء (Part number) . والحقل رقم (٢) هو حقل الكمية الواردة وهو حقل عددي مكون من سبعة أرقام متضمنة رقمين عشريين . والحقل رقم (٣) هو حقل ثمن الشراء وهو حقل عددي مكون من تسعة أرقام متضمنة رقمين عشريين . والحقل رقم (٤) هو حقل تاريخ وصول الصنف وهو حقل تاريخي مكون من ثمانية حروف . والحقل رقم (٥) هو حقل اسم البائع (Vendor) القائم بالتوريد وهو حقل حرفي مكون من خمسين حرفا . والحقل رقم (٦) هو حقل الترحيل وهو حقل منطقي (Logical) يستخدم للتحكم في ترحيل بيانات الصنف الى الملف الرئيسي ، وضمان عدم تكرار ترحيلها .

ولانشاء ملف الفهرس الخاص بهذا الملف يتم كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) .

```
USE Newstock
INDEX ON Part_no TO Newstock
```

وهكذا يتكون النظام حتى الآن من ستة ملفات . ثلاثة منها ملفات قواعد بيانات (Database files) . والثلاثة الآخرين ملفات فهرس (Index files) .

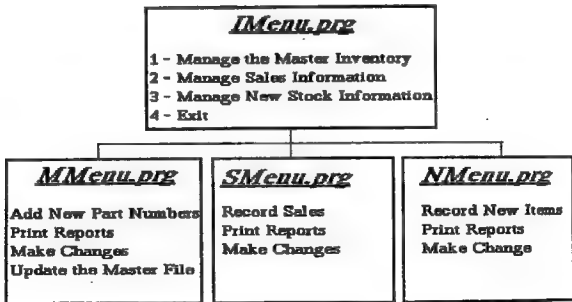
١٢ - ٦ تصميم البرنامج

بعد تصميم ملفات قواعد البيانات كما سبق الايضاح يأتي دور البرنامج . وهو في الواقع ليس برنامجا واحدا ولكنه عدة برامج كما سيتم الايضاح . وهذه البرامج تهدف الى السيطرة على ثلاثة نظم مستقلة ولكنها مرتبطة فيما بينها . النظام الأول هو نظام ادارة المخزون الحالي أو الفعلي ، والذي يكون مسئولاً عن الحالة الفعلية للأصناف الموجودة بالمخزن ومتابعة حالتها وتحديد الأصناف التي تصل الى حد الطلب

(Reorder Point) وهكذا . والنظام الثاني هو حركة المبيعات التي يتم ادخالها بواسطة الموظف المختص في نقطة البيع (Point of sale) . والنظام الثالث هو حركة الاضافة (Newstock) التي يتم فيها تسجيل الأصناف الواردة الى المخزن بواسطة الموظف المختص .

لذلك فان البرنامج يمكن تقسيمه بصفة مبدئية الى أربعة برامج . احدها برنامج رئيسي يقوم بتشغيل البرامج الثلاثة الأخرى . والبرامج الثلاثة الأخرى تقوم بتشغيل برامج فرعية أخرى سيتم دراستها فيما بعد . ولكن سيتم التركيز في هذه المرحلة على هذه البرامج الأربعة والتي تتضح من الشكل (١٢ - ٢) .

وبلاحظ من الشكل أن البرنامج الرئيسي (IMenu.prg) هو البرنامج الذي يحتوي على القائمة الرئيسية التي يتم من خلالها تشغيل البرامج الأخرى . وبناء على اختيار المستخدم يتم التفرع الى البرنامج (MMenu.prg) الذي يشغل الملف الرئيسي للأصناف (Master File) لتتابع الأصناف الموجودة في المخزن . أو يتم التفرع الى البرنامج (SMenu.prg) الذي يشغل ملف المبيعات (Sales File) لتتابع موقف الأصناف التي يتم بيعها . أو يتم التفرع الى البرنامج (NMenu.prg) الذي يشغل ملف الاضافة (Newstock file) لتتابع موقف الأصناف التي يتم توريدها .



شكل (١٢ - ٢)

الفصل الثالث عشر

برنامج القائمة الرئيسية

برنامج القائمة الرئيسية (IMenu.prg) هو البرنامج الذي يقوم بتشغيل القائمة الرئيسية التي يختار المستخدم من خلالها أحد النظم الثلاثة كما سبق الايضاح . وهو لا يختلف في تركيبه عن أى برنامج رئيسي يتم من خلاله عرض قائمة اختيارات (Menu choices) . وكالعادة قبل كتابة أى برنامج يفضل كتابة الخطوات الأولية (PSEUDOCODE) . بأى لغة يجيدها مخطط البرامج . حتى يمكن بعد ذلك كتابة الأوامر بلغة برامج عائلة (DBase) التي تنفذ هذه الخطوات .

١٣ - ١ كتابة الخطوات الأولية (PSEUDOCODE)

يقوم البرنامج الرئيسي في البداية بإنشاء متغير ذاكرة تاريخي اسمه (T_date) ويتم تخزين تاريخ اليوم الحالي (DATE()) في هذا المتغير . ثم يتم عرض هذا التاريخ وسؤال المستخدم إذا كان هذا هو التاريخ السليم أم لا . وفي حالة اختلاله يقوم المستخدم بتعديل التاريخ . وهذه الخطوة مهمة جداً لأن هذا المتغير (T_date) سيتم استخدامه في مواضع متفرقة داخل البرنامج . ثم يقوم البرنامج بسؤال المستخدم عن الاختيار المطلوب من القائمة التي تظهر أمامه . وبناء على اختيار المستخدم يتم التفرع إلى البرنامج الذي يحقق هذا الاختيار .

والخطوات الأولية للبرنامج (PSEUDOCODE) يتم كتابتها كالآتي :

- ١ - مسح كل متغيرات الذاكرة
- ٢ - تجهيز بيئة البرنامج (environment)
- ٣ - مسح الشاشة
- ٤ - إنشاء متغير ذاكرة لتاريخ اليوم الحالي
- ٥ - تكوين حلقة تكرارية لعرض القائمة الرئيسية للبرنامج
- ٦ - مسح الشاشة وعرض القائمة الرئيسية
- ٧ - استقبال اختيار المستخدم
- ٨ - التفرع إلى البرنامج المطلوب
- ٩ - استمرار الحلقة التكرارية حتى يختار المستخدم الخروج
- ١٠ - الخروج من برنامج (DBase III+)

١٣ - ٢ كتابة البرنامج

يتم كتابة البرنامج كما سبق الايضاح بكتابة (MODIFY COMMAND) يليه اسم البرنامج وهو (IMenu) . وغير مطلوب اضافة الامتداد في هذه الحالة لأن البرنامج يضيف الامتداد (.prg) ألياً . و البرنامج يتيح استخدام الحروف الأربعة الأولى فقط من كل أمر . فمثلاً يمكن كتابة (MODI COMM) وهذا يوفر وقتاً كبيراً

عند تعديل البرنامج عدة مرات . ويتم كتابة أوامر البرنامج كالآتي :

```
..... IMenu.prg
*   Main menu for the inventory system
*
SET STATUS OFF
CLEAR ALL
SET TALK OFF
SET BELL OFF
SET SAFETY OFF
SET HEADING OFF
CLEAR

* - - - - - Create underline variable Uline
Uline = REPLICATE ("_", 80)

* - - - - - Create memory variable for today's date
T_Date = DATE()
@ 17,5 SAY "To change date type new date and press" +
"Return "
@ 15,5 SAY "Today's date =" GET T_Date PICT "99/99/99"
READ

* - - - - - Set up loop for presenting main menu.
IChoice = 0
DO WHILE IChoice # 4
  CLEAR
  @ 2,1 SAY 'Inventory system main menu'
  @ 2,60 SAY DTOC(T_Date) + " " + TIME()
  @ 3,0 SAY Uline
  ?
  ?
  TEXT
    1. Manage master inventory
    2. Record sales
    3. Record new stock
    4. Exit
  ENDTXT
```



```
* - - - - - Wait for answer
@ 24,1 SAY "Enter choice : " GET IChoice PICT "9";
RANGE 1,4
READ
DO CASE
    CASE IChoice = 1
        DO MMenu
    CASE IChoice = 2
        DO SMenu
    CASE IChoice = 3
        DO NMenu
ENDCASE
ENDDO(while IChoice # 4)
* - - - - - When done , exit
CLEAR
QUIT
```

والبرنامج يبدأ كالعادة بكتابة اسم البرنامج ووظيفته . ثم يتم تجهيز بيئة البرنامج عن طريق مجموعة من أوامر (SET) وهي كالآتي :

```
SET STATUS OFF
CLEAR ALL
SET TALK OFF
SET BELL OFF
SET SAFETY OFF
SET HEADING OFF
CLEAR
```

وللتعرف على وظيفة كل من هذه الأوامر يمكن الرجوع الى الكتاب الثانى .

والجزء الثانى من البرنامج يبدأ بإنشاء متغير ذاكرة (Uline) يحتوى على حرف الشرطة السفلية (_) مكرراً ثمانين مرة . وهذا يؤدي الى تخزين سطر بعرض الشاشة يمكن عرضه فى أى مكان بعد ذلك . مع ملاحظة أن إنشاء هذا المتغير فى البرنامج الرئيسى يجعله عاماً (Public) بالنسبة للبرامج الفرعية الأخرى . أى يمكن استخدامه مباشرة فى أى برنامج فرعى . وهذا الأمر يظهر فى البرنامج كالآتي :

```
Uline = REPLICATE ( " _ " , 80 )
```

والجزء الثالث من البرنامج يبدأ بإنشاء متغير الذاكرة التاريخي (T_DATE) وتخزين تاريخ اليوم الحالي فيه ، وهو التاريخ الذي يتم ادخاله عند تشغيل الجهاز من خلال نظام التشغيل (MS-DOS) . وإذا لم يكن المستخدم قد أدخل التاريخ لي بداية تشغيل الجهاز ، فإن البرنامج يعرض له التاريخ السابق تخزينه في المتغير (T_Date) . ويستطيع في هذه الحالة تعديل هذا التاريخ ليوافق تاريخ اليوم الحالي . وهذه العملية تتم من خلال مجموعة السطور التالية :

```
T_Date = DATE()
@ 17,5 SAY "To change date type new date and" + ;
"press Return "
@ 15,5 SAY "Today's date =" GET T_Date PICT "99/99/99"
```

والمتغير (T_date) مهم جداً لأنه سوف يستخدم في ملء بيانات الحقول التاريخية في ملفات المبيعات والأضافة آلياً .

والجزء الرابع يقوم بتكوين الحلقة التكرارية لمسح الشاشة وعرض العنوان متضمناً التاريخ والوقت ثم خط أسفل هذا العنوان ثم عرض القائمة الرئيسية السابق شرحها ، ثم يتم عرض سؤال للمستخدم عن الاختيار المطلوب واستقبال هذا الاختيار في المتغير (IChoice) . وهذا يتم من خلال السطور التالية :

```
IChoice = 0
DO WHILE IChoice # 4
    CLEAR
    @ 2,1 SAY "Inventory system main menu"
    @ 2,60 SAY DTOC(T_Date) + " " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT
        1. Manage master inventory
        2. Record sales
        3. Record new stock
        4. Exit
    ENDTXT
    * - - - - - Wait for answer .
    @ 24,1 SAY "Enter choice:" GET IChoice PICT "9" ;
    RANGE 1,4
    READ
```

والجزء الخامس من البرنامج يستعمل الأمر (DO CASE) في التفرع الى البرنامج المطلوب حسب اختيار المستخدم . وهذا يتم من خلال السطور التالية :

```
DO CASE
  CASE IChoice = 1
    DO MMenu
  CASE IChoice = 2
    DO SMenu
  CASE IChoice = 3
    DO NMenu
ENDCASE
```

وفي الجزء الأخير من البرنامج يتم إنهاء الحلقة التكرارية ومسح الشاشة والرجوع الى نظام التشغيل . وذلك كالآتي :

```
ENDDO(while IChoice # 4)
* - - - - - When done , exit
CLEAR
QUIT
```

١٣ - ٢ اختيار البرنامج

بعد الانتهاء من كتابة البرنامج يتم اختباره وذلك بتشغيله كالآتي :

```
DO IMenu
```

ثم ملاحظة تنفيذ خطوات البرنامج وظهور الاختيارات على الشاشة . ويمكن ادخال قيم خارج المدى المسموح وهو من (1) الى (4) وذلك بادخال (5) مثلاً أو أى رقم آخر اكبر من (4) ، وملاحظة ما يحدث . وبعد انتهاء اختبار البرنامج يتم العودة الى نظام التشغيل عن طريق الاختيار (4) .

والشكل (١٣ - ١) يوضح شكل الشاشة التي تظهر عند تنفيذ هذا البرنامج .

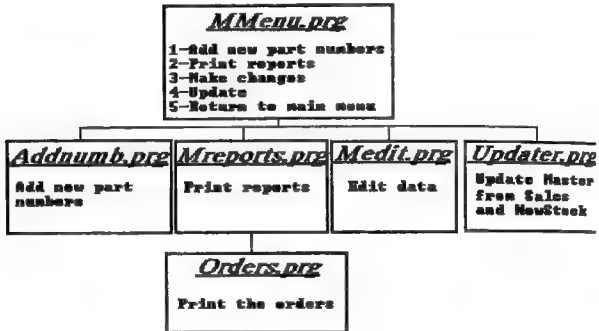
Inventory system main menu	02/20/90	12:40:50
<ol style="list-style-type: none">1. Manage master inventory2. Record sales3. Record new stock4. Exit		
Enter choice		

شكل (١٢ - ١)

الفصل الرابع عشر

برنامج تشغيل الملف الرئيسي

هذا البرنامج يقوم بإدارة وتشغيل المعلومات المخزنة في الملف الرئيسي (Master file) بهدف تحديد الحالة الفعلية (Current status) للأصناف في المخزن . وهذا الملف سبق تكوينه وتسميته (Master.dbf) . كما سبق انشاء الفهرس الخاص به بناء على حقل رقم الجزء (Part_no) . ويتم تشغيل هذا البرنامج عند اختيار الرقم (١) من القائمة الرئيسية . انظر الشكل (١٤ - ١)



شكل (١٤ - ١)

١٤ - ١ تصميم برنامج تشغيل الملف الرئيسي

يتكون برنامج تشغيل الملف الرئيسي من ستة برامج منفصلة . أحدها يمثل البرنامج الرئيسي (Main Program) الذي يتحكم في البرامج الخمسة الأخرى . انظر الشكل (١٤ - ١) .

ويوضح الشكل اسم كل برنامج ثم وظيفة هذا البرنامج ، مع ملاحظة أن البرنامج الرئيسي (MMenu.prg) وظيفته الرئيسية عرض القائمة المبينة والتفرع الى كل برنامج من البرامج الفرعية بناء على اختيار المستخدم .

١٤ - ٢ تصميم البرنامج الرئيسي

البرنامج الرئيسي في هذه الحالة لا يختلف عن برامج القائمة الرئيسية السابق شرحها . لذلك ليست هناك حاجة لشرحه بالتفصيل . ونكتفي هنا بكتابة البرنامج كالآتي :

```
..... MMenu.prg
*   Menu for master program of inventory system
*   Called from inventory system main menu
* - - - - - Set up loop for presenting menu
Mchoice = 0
DO WHILE Mchoice # 5      CLEAR
    @ 2,1 SAY "Manage Master Inventory
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
TEXT

    1. Add new part numbers
    2. Print reports
    3. Make changes
    4. Update from Sales and Newstock
    5. Return

ENDTEXT
@ 24,1 SAY "Enter choice(1-5)" ;
GET Mchoice PICT "9" RANGE 1,5
READ
DO CASE
CASE Mchoice = 1
DO Addnumbs
CASE Mchoice = 2
DO Mreports
CASE Mchoice = 3
DO Medit
CASE Mchoice = 4
```



```
DO Updater
ENDCASE
ENDDO(While Mchoice # 5)

* - - - - - when done , return to main menu.
RETURN
```

ومند تنفيذ هذا البرنامج تظهر الشاشة الموضحة في الشكل (١٤ - ٢)

Manage Master Inventory	02/20/90	12:40:50
<ul style="list-style-type: none">1. Add new part numbers2. Print reports3. Make changes4. Update from sales and Newstock5. Return to main menu		
Enter choice (1 - 5)		

شكل (١٤ - ٢)

١٤ - ٣ برنامج اضافة الأصناف

هذا البرنامج هو البرنامج الذي يتم التفرع اليه عند اختيار المستخدم للرقم (١) في القائمة السابقة . في هذه الحالة يتم تنفيذ البرنامج (Addnumbs.prg) . وقبل كتابة البرنامج يجب أولا انشاء شاشة الادخال التي سوف تستخدم في ادخال بيانات الصنف الجديد .

١٤ - ٣ - ١ انشاء شاشة الادخال

يتم انشاء شاشة الادخال باحدى طريقتين ، الطريقة الأولى باستخدام راسم الشاشة (Screen Painter) الذي يظهر عن طريق قوائم المساعد (Assistant) أو عن طريق كتابة الأمر (CREATE SCREEN) من خلال مشيرة النقطة (Dot Prompt) . والطريقة الثانية من طريق كتابة ملف أوامر (Command File) . وسنستخدم الطريقة الأولى في هذا البرنامج .

ولتنفيذ ذلك يتم كتابة السطر التالي عند مشيرة النقطة (Dot Prompt) :

CREATE SCREEN Iscreen1

وعند ظهور عمود الاختيارات على الشاشة يتم اختيار (Database File) . ومن خلاله يتم اختيار ملف قاعدة البيانات (Master.dbf) ، ثم يتم اختيار (Load Fields) . حيث يتم اختيار الحقول المطلوب ظهورها في الشاشة . ويتم اختيار كل حقل عن طريق تحريك العمود الضوئي الى هذا الحقل والضغط على مفتاح الادخال . في هذه الحالة يظهر مثلث (٥) أمام اسم الحقل . ويمكن اختيار كل الحقول ما عدا حقل تاريخ آخر طلب للصنف (Order_Date) وحقل الكمية التي يجب طلبها من الصنف (New_Order) لأن هذين الحقلين يتم ملأهما آلياً من خلال البرنامج . وعند الانتهاء من ادخال الحقول يتم الضغط على مفتاح (<-->) للخروج من هذه القائمة ولعرض السبورة (Blackboard) التي نسيتم من خلالها تحديد مواضع الحقول حسب الحاجة .

وعند الانتهاء من تحديد مواضع الحقول على الشاشة يتم تحريك المؤشر الى العمود الضوئي الخاص برقم الجزء (Part number) ويتم تحويل حالة هذا الحقل من (Edit Get) الى (Display Say) . وذلك لكي يصبح هذا الحقل غير قابل للتعديل بواسطة المستخدم .

ولتنفيذ ذلك يتم الضغط على مفتاح (F10) والضغط على مفتاح الادخال عند الاختيار (ACTION) لتحويله الى (DISPALY SAY) . ولاحظ في هذه الحالة اختفاء العمود الضوئي الخاص برقم الجزء من الشاشة .

ولمزيد من التفاصيل عن تصميم شاشة الادخال يمكن الرجوع الى الكتاب

- الأول . والشكل رقم (١٤ - ٢) يوضح تصميم مقترح لشاشة الادخال . ويمكن تصميم أى شكل آخر حسب الحاجة .

Master Inventory File	
Part number :	Date :
Part name :	Unit cost :
Quantity in stock :	Reorder point :
Quantity on order :	
Storage location :	
Vendor	Name
	Address

شكل (١٤ - ٢)

١٤ - ٢ - ٢ كتابة الخطوات الأولية (PSEUDOCODE)

بعد تصميم الشاشة يتم كتابة الخطوات الأولية لبرنامج (Addnumbs.prg) وذلك كالآتي :

- ١- يتم فتح الملف الرئيسي (Master.dbf) وملف الفهرس الخاص به .
- ٢- يتم تكوين حلقة تكرارية لاضافة الأصناف الجديدة .
- ٣- يتم مسح الشاشة .
- ٤- يتم سؤال المستخدم عن رقم الصنف المطلوب اضافته .
- ٥- يتم اختبار هذا الرقم للتأكد أنه لم يسبق ادخاله في الملف .
- ٦- اذا لم يتم ادخال رقم الصنف يتم الرجوع الى القائمة الرئيسية .
- ٧- اذا كان الرقم موجوداً يتم تنبيه المستخدم لاعادة المحاولة مرة ثانية .
- ٨- اذا كان الرقم غير موجود يتم اضافته وفتح شاشة الادخال لادخال بيانات هذا الصنف الجديد .
- ٩- يتم استمرار تنفيذ الحلقة التكرارية حتى يضغط المستخدم على مفتاح

- الادخال دون ادخال رقم صنف جديد .
- ١٠- يتم الرجوع الى القائمة الرئيسية .

١٤ - ٣ - ٣ كتابة برنامج الاضافة (Addnums.prg)

يتم كتابة الامر (MODI COMM Addnums) من مشيرة النقطة ثم
يتم كتابة سطور البرنامج كالآتي :

```
.....Addnums.prg
*   Add new items to the Master file
*   Called from MMenu.prg

USE Master INDEX Master

* - Set up loop for adding new part numbers
Partnumb = "X"
DO WHILE Partnumb # " "
    CLEAR
    @ 2, 1 SAY "Add New Part Numbers"
    @ 2,60 SAY DTOC(T_Date) + " " + TIME()
    @ 3, 0 SAY Uline
    ?
    ?
    * - - - - GET proposed part number
    Partnumb = SPACE(6)
    @ 15,6 SAY "Enter part number(or press " + ;
    " Return to exit)" GET Partnumb
    READ

    * - Check to see if part number already exists
    partnumb = UPPER(partnumb)
    SEEK partnumb
    DO CASE
        CASE Partnumb = " "
            CLEAR
    CASE FOUND()
        @ 20,10 SAY Partnumb + "already exists"
```

```

? CHR(7)
WAIT 'Press any key to try again'
CASE .NOT. FOUND()
  APPEND BLANK
  REPLACE Part_no WITH Partnumb
  REPLACE Date WITH T_Date
  SET FORMAT TO Iscreen1
  READ
  SET FORMAT TO
ENDCASE
ENDDO(While partnumb = " " )
* - - - - - Return to Master menu
RETURN

```

والجزء الأول من البرنامج يبدأ كالمادة بكتابة اسم البرنامج ووظيفته ثم تحديد البرنامج القائم باستخدامه . بعد ذلك يتم فتح ملف قاعدة البيانات (Master.dbf) وملف الفهرس المقابل (Master.ndx).

والجزء الثاني من البرنامج يتم من خلاله تكوين الحلقة التكرارية بعد ادشاء المتغير الحرفي (Partnumb) واعطائه القيمة (X). والقيمة (X) في هذه الحالة تضمن تنفيذ الحلقة التكرارية على الأقل مرة واحدة . ثم يبدأ تنفيذ الحلقة التكرارية بمسح الشاشة وعرض عنوان (Header) مع عرض التاريخ والوقت ثم يتم سؤال المستخدم عن رقم الجزء المطلوب اضافته وتخزين هذا الرقم في المتغير (Partnumb). ويجب ملاحظة أن الضغط على مفتاح الادخال دون كتابة رقم الجزء يؤدي الى الخروج من الحلقة التكرارية والرجوع الى القائمة الرئيسية . وهذا الجزء يتكون من السطور التالية :

```

Partnumb = "X"
DO WHILE Partnumb # " "
  CLEAR
  @ 2, 1 SAY "Add New Part Numbers"
  @ 2,60 SAY DTOC(T_Date) + " " + TIME()
  @ 3, 0 SAY Uline
  ?
  ?
  * - - - - - GET proposed part number
  Partnumb = SPACE(6)
  @ 15,6 SAY "Enter partnumb(or press " + ;

```

```
" Return to exit)" GET Partnumb
READ
```

والجزء الثالث من البرنامج يبدأ بتحويل رقم الجزء الى حروف كبيرة (Uppercase) ثم البحث عن رقم الجزء في الملف الرئيسي . وحيث أن الملف مفهرس ، لذلك يستخدم الأمر (SEEK) في البحث عن هذا الرقم .

ويحتوى هذا الجزء على السطور التالية :

```
Partnumb = UPPER(partnumb)
SEEK Partnumb
```

والجزء الرابع من البرنامج يختص باتخاذ القرار بناء على ما يدخله المستخدم . وفي الحالة الأولى عندما يكون (Partumb = " ") ، فإن هذا يعنى أن المستخدم يضغط على مفتاح الإدخال دون كتابة أى رقم جزء (Partnumb) . وفي هذه الحالة يتم مسح الشاشة ثم تنفيذ الأمر الذى يلي الأمر (ENDCASE) . وهذا يؤدي الى الخروج من الحلقة التكرارية لأن الشرط الموجود في أول الحلقة التكرارية لا يتحقق .

وفي الحالة الثانية ((CASE FOUND)) ، فإن هذا يعنى أن هذا الجزء قد سبق ادخاله في الملف . لذلك يتم تنبيه المستخدم بعرض رسالة توضح له أن هذا الرقم موجود . وذلك بالإضافة الى تشغيل الجرس للتنبيه مع استخدام الأمر (WAIT) في عرض الرسالة الموضحة وانتظار ضغط المستخدم على أى مفتاح حتى يمكنه ادخال رقم جديد .

وفي الحالة الثالثة ((CASE .NOT. FOUND)) ، فإن هذا يعنى أن المستخدم أدخل رقماً غير موجود داخل الملف . وفي هذه الحالة فإن البرنامج يعرض شاشة الإدخال ثم يسمح للمستخدم بإدخال باقى بيانات الصنف مع عدم السماح له بكتابة رقم الجزء من خلال شاشة الإدخال . وإنما يتم ادخال رقم الجزء بواسطة البرنامج عن طريق الأمر (REPLACE) . كما يتم ادخال تاريخ آخر تعديل (Date) عن طريق البرنامج أيضا .

ويتكون هذا الجزء من السطور التالية :

```
DO CASE
CASE Partnumb = " "
CLEAR
CASE FOUND()
```

```
@ 20,10 SAY Partnumb + "already exists"
? CHR(7)
WAIT 'Press any key to try again'
CASE .NOT. FOUND()
APPEND BLANK
REPLACE Part_no WITH Partnumb
REPLACE Date WITH T_Date
SET FORMAT TO Iscreen1
READ
SET FORMAT TO
ENDCASE
```

والجزء الأخير من البرنامج يحتوى على أمر إنهاء الحلقة التكرارية والعودة الى القائمة الرئيسية الخاصة بالملف الرئيسي (MMenu) . وإذا لم يرد المستخدم الخروج فان الحلقة التكرارية تستمر . أما اذا أراد الخروج فانه يضبط على مفتاح الادخال دون ادخال اى رقم ، وفي هذه الحالة يتم الرجوع الى القائمة الرئيسية .

١٤ - ٤ برنامج تقارير الملف الرئيسي

يمكن تصميم تقارير الملف الرئيسي (Master.dbf) من طريق قوائم برنامج المساعد (Assistant) أو باستخدام الأمر (CREATE REPORT) . وفي الحالتين تظهر اختيارات تصميم التقرير التي يتم عن طريقها ادخال المعاملات المختلفة التي توضع عنوان الصفحة (Page Title) وأسماء الحقول بالاضافة الى المعاملات الأخرى . وكما سبق الايضاح فان هناك طريقة أخرى لتصميم التقرير عن طريق ملف الأوامر (Command file) . وهي تتيح امكانيات أكبر في تحديد أماكن الحقول وشكل التقرير بصفة عامة . وبالنسبة للملف الرئيسي (Master.dbf) فسوف نعرض للطريقتين في انشاء التقارير الخاصة به .

وهناك أربعة تقارير مطلوبة للملف الرئيسي يتم شرحها في الأجزاء التالية :

١٤ - ٤ - ١ تقرير المخزون الحالي (Current Stock)

هذا التقرير يوضح الموقف الحالي للأصناف في المخزن من حيث اسم كل صنف والكمية الفعلية الموجودة منه ومكانه في المخزن وسعر الوحدة وهكذا . ويتم انشاء هذا التقرير عن طريق كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) .

USE Master
CREATE REPORT Allmast

حيث (Allmast) هو اسم ملف التقرير . ويلاحظ في هذه الحالة ظهور قوائم تصميم التقرير السابق شرحها في الجزء الأول ولن نتعرض لها هنا بالتفصيل ولكن سيتم توضيح المعاملات التي يجب ادخالها حتي نحصل على شكل التقرير المطلوب . والشكل التالي يوضح قيم المعاملات الخاصة بصفحة التقرير .

Pagetype	Current stock
Page width	74
Left margin	1
Right margin	0
Lines per page	52
Double space report	NO

شكل (١٤ - ٤)

كما يتم تحديد محتويات أعمدة التقرير (Columns) كالآتي :

Column No.	Contents	Heading	Width	Decimals	Total
1	Part_no	Part No	6		
2	Title	Part Name	20		
3	Qty	On Hand	7	2	N
4	Cost	Unit Cost	9	2	N
5	Reorder	Reorder	8	0	N
6	Location	Location	9		
7	Date	Last Update	9		

شكل (١٤ - ٥)

ولتوضيح شكل التقرير بعد طباعته نفرض بيانات بعض الأصناف؛ ثم نكتب الأمر (REPORT FORM Allmast). وفي هذه الحالة يظهر التقرير كالآتي مثلاً :

Page No		Current Stock			
102/20/90					
Part No.	Part Name	On Hand	Unit Cost	Reorder Location	Last Update
BBB	Floppy Disk	40	14	10-A-111	02/15/90
AAA	Printer	10	800	17-B-233	02/10/90
ZZZ	Bicycle	10	80	19-G-175	01/01/90

شكل (١٤ - ٦)

١٤ - ٤ - ٢ تقرير حد الطلب (Reorder)

وهو التقرير الذي يعرض بيانات الأصناف التي تقل عن حد الطلب . ويتم انشاؤه عن طريق الأمر (CREATE REPORT Reorders) كما سبق الايضاح . ويتم استخدام المعاملات الموضحة في الشكل التالي :

Page Title	Goods to be Reordered
Page width	77
Left margin	1
Right margin	0
Lines per page	58
Double space report	NO

شكل (١٤ - ٧)

كما يتم ادخال محتويات اعمدة التقرير (Columns) كالآتي :

Column No	Contents	Heading	Width	Decimals	Total
1	Part_no	Part No	6		
2	Title	Part Name	20		
3	Qty	On Hand	7	2	N
4	Cost	Unit Cost	9	2	N
5	Reorder	Reorder	8	0	N
6	Location	Location	9		
7	Date	Last Update	8		

شكل (١٤ - ٨)

ولتوضيح شكل التقرير عند طباعته يتم كتابة الأمر (REPORT FORM Reorders) في هذه الحالة يظهر التقرير الآتي :

Page No. 1 02/20/90					
<u>Goods to be Reordered</u>					
Part no	Part name	On Hand	On Order	reorder	Vendor Name
AAA	Printer	5	5	10	Micronet Company

شكل (١٤ - ٩)

١٤ - ٤ - ٣ تقرير الأصناف تحت الطلب

ويتم انشاء هذا التقرير بنفس الطريقة مثل التقارير السابقة من طريق الأمر (CREATE REPORT Onorder) ، مع ادخال المعاملات الواضحة في

الشكل التالي :

Page title	Items Currently On Order
Page Width	77
Left Margin	1
Right Margin	0
Lines per Page	58
Double space reprot	NO

شكل (١٤ - ١٠)

كما يتم ادخال بيانات الأعمدة بالاستعانة بالبيانات الموجودة في الشكل التالي :

Item No	Description	Quantity	Unit Cost	Total Cost	Part
1	Order_date	Order Date	8		
2	Part_no	Part No	6		
3	Title	Part Name	15		
4	On_order	On Order	7	2	N
5	Cost	Unit Cost	9	2	N
6	Cost*On_order	Total Cost	9	2	Y
7	Vendor	Vendor Name	25		

شكل (١٤ - ١١)

ولعرض التقرير يتم كتابة الأمر (REPORT FORM Onorder) .

وفي هذه الحالة يظهر التقرير كالاتي مثلا :

Page No. 1		Item Quantity On Order			
12/22/90					
Order Date	Part No	Part Name	On order	Unit Cost	Total Vendor Name
01/02/90	AAA	Printer	5	800	4000 Micronet Company
TOTAL					4000

شكل (١٤ - ١٢)

وبلاحظ هنا أن التقرير يقوم بتجميع البيانات الموجودة في عمود التكلفة الكلية (Total Cost) ، وذلك لأنه قد سبق ادخال الاختيار (Y) في العمود (Total?) كما هو واضح من الشكل الخاص ببيانات الأعمدة .

١٤ - ٤ - طلب الشراء (Purchase Order)

طلب الشراء هو أحد التقارير التي يتم الحصول عليها من خلال برنامج تشغيل الملف الرئيسي (Master.dbf) . والبرنامج يعرض على المستخدم بيانات الأصناف التي تصل إلى حد الطلب ويتيح للمستخدم طلب العدد الذي يريده من هذه الأصناف . حيث يعرض البرنامج الشاشة التالية لكل صنف من الأصناف التي تصل إلى حد الطلب .

Part number	D-100	Sub
On hand	30	
On order	10	
Reorder	50	
Unit cost	100	
Order how many?		

شكل (١٤ - ١٢)

ومن خلال هذه الشاشة يستطيع المستخدم أن يلاحظ بسرعة عدد البديل (Suits) الموجودة في المخزن وعدد البديل الجارى طلبها (On Order) وحد الطلب (Reorder). وبناء على ذلك يحدد عدد البديل المطلوب شراؤها لتعويض النقص الموجود . ويقوم بادخال هذا العدد أمام السؤال المبين . ويتم تكرار هذه العملية مع باقى الأصناف التى وصلت الى حد الطلب (Reorder) . وبعد الانتهاء من ادخال الأعداد المطلوبة من كل صنف يقوم البرنامج أكيا بطباعة طلبات الشراء لهذه الأصناف . والشكل التالى يوضح نموذجاً لأحد هذه الطلبات .

Emaar Afandy Company
12 - Ahran Street

Please send us the following items...

10	suits	100	1000
5	Printers	800	4000
Total Cost			5000

Mail to **AAA Company**
 12 - Ahran Street

شكل (١٤ - ١٤)

١٤ - ٤ - ٥ تصميم برنامج التقارير

يتكون برنامج التقارير من برنامجين أحدهما هو البرنامج (MReports.prg) الذى يعرض القائمة الرئيسية للتقارير ، وهى القائمة التى يختار المستخدم منها التقرير المطلوب طباعته . والبرنامج الآخر هو برنامج طلبات الشراء (Orders.prg) الذى يتم عن طريقه طباعة طلبات الشراء للأصناف التى تقل عن حد الطلب فى المخزن .

١٤ - ٤ - ٦ تصميم برنامج القائمة

وهذا البرنامج لا يختلف عن برامج القوائم الأخرى . ولكن يجب ملاحظة أن فتح ملف قاعدة البيانات وملف الفهرس المرتبط به يتم في بداية البرنامج وقبل الدخول في الحلقة التكرارية . وذلك لأن الملفين يستخدمان في جميع التقارير بعد ذلك . والبرنامج يتكون من السطور التالية :

```
..... MReports.prg
*   Present report options for Master file
*   Called from Master menu , MMenu.prg
USE Master INDEX Master
* - - - - Set up loop for presenting menu
Repchoice = 0
DO WHILE Repchoice # 5
    CLEAR
    @ 2,1 SAY "Master Inventory Report Options"
    @ 2,60 SAY DTC(T_Date) + " " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT

        1. Entire inventory
        2. Reorder report
        3. On Order report
        4. Purchase orders
        5. Return to master menu

    ENDTEXT
    @ 24,1 SAY "Enter choice(1-5)" ;
    GET Repchoice PICT "9" RANGE 1,5
    READ
    * - - If not choosing Purchase orders, ask about
    * - - printer .
    CLEAR
    STORE " " TO YN, Printer
    IF Repchoice < 4
```

```

@ 5,5 SAY "Send report to printer ?" ;
GET YN PICT "!"
READ
CLEAR

* - - - - Set up for printer.
IF YN = "Y"
    Printer = "TO PRINT"
ENDIF
ENDIF(Repchoice < 4)
DO CASE
    CASE Repchoice = 1
        REPORT FORM Allmast & Printer
    CASE Repchoice = 2
        REPORT FORM Reorders FOR (QTY + On_order);
        < = Reorder & Printer
    CASE Repchoice = 3
        REPORT FORM Onorder FOR On_order > 0 ;
        & Printer
    CASE Repchoice = 4
        DO Orders
ENDCASE

* - - - IF report not going to printer and not
* - - - exiting program , pause .
IF YN # "Y" .AND. Repchoice # 5
?
?
WAIT "Press any key to return to the reports menu"
ENDIF
ENDDO (Repchoice # 5)
RETURN

```

والجزء الأول من البرنامج يبدأ بالسطور المعتادة لكتابة اسم البرنامج الذي يقوم باستخدامه . ثم يتم فتح الملف الرئيسي (Master.dbf) وملف الفهرس الخاص به (Master.ndx) . ويتم بعد ذلك تكوين الحلقة التكرارية التي يتم من خلالها عرض قائمة الاختيارات التي يختار المستخدم منها نوع التقرير المطلوب . ويتكون هذا الجزء من السطور التالية :

منها نوع التقرير المطلوب . ويتكون هذا الجزء من السطور التالية :

```
USE Master INDEX Master
Repchoice = 0
DO WHILE Repchoice # 5
CLEAR
  @ 2,1 SAY "Master Inventory Report Options"
  @ 2,60 SAY DTOC(T_Date) + " " + TIME()
  @ 3,0 SAY Uline
  ?
  ?
  TEXT
    1. Entire inventory
    2. Reorder report
    3. On Order report
    4. Purchase orders
    5. Return to master menu
  ENDTXT
  @ 24,1 SAY "Enter choice(1-5)" ;
  GET Repchoice PICT "9" RANGE 1,5
  READ
```

والاختيارات الثلاثة الأولى تستخدم التقارير التي سبق انشاؤها في عرض البيانات المطلوبة . أما الاختيار رقم (4) فانه يؤدي الى تشغيل البرنامج (Orders.prg) . وهذا البرنامج يعرض على المستخدم بيانات الأصناف التي تقل عن حد الطلب (Reorder) ، ويتيح له تحديد الكمية المطلوبة من كل صنف . ثم يقوم البرنامج بطباعة أوامر الشراء كليا .

والجزء الثاني يبدأ بإنشاء متغيرات الذاكرة (YN) ، (Printer) واعطائهما القيمة (" ") . ثم سؤال المستخدم إذا كان يريد طباعة التقرير على الطابعة أو عرضه على الشاشة ، وذلك بالنسبة للاختيارات التي تقل عن (4) . وذلك لأن الاختيار (4) يتفرع الى برنامج (Orders.prg) الذي يقوم بطباعة أوامر الشراء (Purchase orders) .

وعندما يختار المستخدم الاختيار (Y) فإن الشرط الموجود بعد (IF) يتحقق . ويؤدي هذا الى تخزين العبارة (TO PRINT) في المتغير (Printer) . ويتكون هذا الجزء من السطور التالية :


```

CLEAR
STORE " " TO YN, Printer
IF Repchoice < 4
    @ 5,5 SAY "Send report to printer ?" ;
    GET YN PICT "!"
    READ
    CLEAR

    * - - - - Set up for printer.
    IF YN = "Y"
        Printer = "TO PRINT"
    ENDIF
ENDIF(Repchoice < 4)

```

والجزء الثالث يتم فيه اتخاذ القرار تبعاً لاختيار المستخدم . فعندما يختار الرقم (1) يتم تكوين التقرير (Allmast) الذي سبق أنشأه . ويلاحظ استخدام دالة الماكرو (&) للتمويض عن المتغير (Printer) بالقيمة المخزنة فيه ، وهي (TO PRINT) . أى أن الأمر يصبح كالآتي :

REPORT FORM Almast TO PRINT

وهذا يؤدي الى طباعة التقرير متضمنا بيانات الأصناف الموجودة بالمخزن .

وعندما يختار المستخدم الرقم (2) يتم تكوين التقرير (Reorders) الذي سبق أنشأه . ويلاحظ أن الأمر المستخدم فى هذه الحالة يستخدم شرطاً لتحديد السجلات التي تظهر فى التقرير . وهي سجلات الأصناف التي تقل كميتها عن حد الطلب .

وعندما يختار المستخدم الرقم (3) يتم طباعة التقرير (Onorders) ، وذلك بالنسبة للأصناف التي توجد منها كميات تحت الطلب فقط . وذلك عن طريق استخدام الشرط (On_order > 0) .

واستخدام دالة الماكرو (&) فى الاختيارات الثلاث السابقة يفيد فى التحكم فى طباعة التقرير أو عرضه على الشاشة حسب اختيار المستخدم . فإذا أراد المستخدم طباعة التقرير فإنه يدخل الحرف (Y) فى المتغير (YN) . وهذا يؤدي الى ادخال العبارة (TO PRINT) فى المتغير

(printer) كما سبق الايضاح . وهذا بالتالى يؤدي الى ادخال عبارة (TO PRINT) بعد الأمر (REPORT FORM) ، مما يؤدي الى طباعة التقرير .

أما إذا أراد المستخدم عرض التقرير على الشاشة فقط دون طباعته ، فإنه يدخل أى قيمة أخرى غير (Y) فى المتغير (YN) . وبالتالى لايتحقق الشرط بعد (IF) ، ويظل المتغير (printer) خالياً . وهذا يؤدي الى عدم اضافة أى عبارة بعد الأمر (REPORT FORM) . وبالتالى يتم عرضه على الشاشة فقط .

وعندما يختار المستخدم الاختيار (4) ، فإن البرنامج يتفرع الى البرنامج الفرعى (Orders) الذى سيتم دراسته فيما بعد .

والجزء الثالث الذى سبق شرحه يتكون من السطور التالية :

```
DO CASE
CASE Repchoice = 1
REPORT FORM Allmast & Printer
CASE Repchoice = 2
REPORT FORM Reorders FOR (QTY + On_order) ;
< = Reorder & Printer
CASE Repchoice = 3
REPORT FORM Onorder FOR On_order > 0 ;
& Printer
CASE Repchoice = 4
DO Orders
ENDCASE
```

والجزء الرابع والأخير من البرنامج يتم من خلاله ايقاف الشاشة مؤقتا فى حالة اختيار المستخدم عرض التقرير على الشاشة وليس على الطابعة . وذلك حتى يستطيع المستخدم قراءة بيانات التقرير على الشاشة . ثم يقوم بالضغط على أى مفتاح للرجوع الى قائمة التقارير مرة أخرى حتى يختار نوعا آخر من التقارير حسب الحاجة أو يختار الرقم (5) للخروج من برنامج التقارير والعودة الى القائمة الرئيسية لبرنامج تشغيل الملف الرئيسى (MMenu.prg) .

١٤ - ٤ - ٧ برنامج أوامر الشراء

لعرض أو طباعة أوامر الشراء (Purchase orders) فإن برنامج التقارير (MReports.prg) يتفرع الى برنامج أوامر الشراء

(orders.prg) . وهذا البرنامج يؤدي عدة وظائف . فهو في البداية يجب أن يبحث خلال الملف الرئيسي (Master.dbf) عن الأصناف التي يجب طلبها . ولتنفيذ ذلك فإنه يجمع الكمية الموجودة فعلا (On hand) على الكمية تحت الطلب (On order) ويقارن المجموع بحد الطلب (Reorder) . وفي كل مرة يجد فيها صنفا يقل عن حد الطلب ، فإنه يسأل المستخدم عن الكمية المطلوب شراؤها من هذا الصنف . وبعد الانتهاء من ادخال كل الأصناف المطلوب شراؤها ، فإنه يطبع أوامر الشراء . وفي نفس الوقت يقوم بتحديث الكمية الموجودة في الحقل (On_order) في الملف الرئيسي ، وذلك بجمع الكمية الجديدة على هذا الحقل بالنسبة لكل صنف يتم ادخاله في أوامر الشراء .

ولتصميم هذا البرنامج نقوم أولا بكتابة الخطوات الأولية (PSEUDOCODE) وهي تكون كالآتي :

- ١ - يتم مسح الشاشة .
- ٢ - يتم فتح ملف قاعدة البيانات وملف الفهرس الخاص به .
- ٣ - يتم تكوين حلقة تكرارية خلال الملف الرئيسي (Master.dbf) .
- ٤ - يتم اختبار مجموع الكمية الفعلية (on hand) مع الكمية تحت الطلب (On order) ومقارنة المجموع بحد الطلب . وذلك بالنسبة لكل صنف من أصناف الملف الرئيسي (Master.dbf) .
- ٥ - يتم مسح الشاشة .
- ٦ - يتم عرض بيانات حالة الصنف الذي يجب طلبه .
- ٧ - يتم سؤال المستخدم عن الكمية المطلوب طلبها من هذا الصنف .
- ٨ - يتم تحديث الحقل (New_order) الذي يمثل الكمية التي يجب طلبها من الصنف وكذلك تحديث الحقل (Order_date) الذي يمثل تاريخ آخر طلب لهذا الصنف .
- ٩ - عند ادخال كل الأصناف التي يجب طلبها يتم مسح الشاشة .
- ١٠ - يتم انشاء ملف مؤقت (Temporary File) للأصناف المطلوبة .
- ١١ - يتم تحديث حقل تحت الطلب (On_order) في الملف الرئيسي (Master.dbf) .
- ١٢ - يتم إعادة محتويات حقل (New_order) الى الصفر .
- ١٣ - يتم فتح الملف المؤقت ، الذي يحتوي على بيانات الأصناف المطلوبة فقط .
- ١٤ - يتم فهرسة هذا الملف بناء على حقل البائع (Vendor) وذلك لتقسيم الأصناف بالنسبة للبائعين .
- ١٥ - يتم تشغيل الطابعة .
- ١٦ - يتم طباعة اسم البائع وعنوانه .
- ١٧ - يتم طباعة الكمية المطلوبة واسم الصنف وسعره لكل صنف من الأصناف

- التي يتم شراؤها من هذا البائع .
- ١٨ - عند الانتهاء من هذا البائع ، يتم طباعة السعر الكلي لهذا الطلب كما يتم طباعة اسم المكان المطلوبة له هذه الأصناف وعنوانه .
- ١٩ - يتم الانتقال الى صفحة جديدة علي الطابعة ، وكذلك الانتقال الى بائع جديد (Vendor) في الملف المؤقت .
- ٢٠ - تستمر هذه العملية بالنسبة لجميع البائعين في الملف المؤقت .
- ٢١ - عند الانتهاء يتم إغلاق الطابعة والعودة الى برنامج التقارير .

١٤ - ٤ - ٨ كتابة البرنامج

بعد كتابة الخطوات الأولية (PSEUDOCODE) للبرنامج يتم كتابة أوامر البرنامج وهي تتلخص في السطور التالية :

```
..... Orders.prg
* Create Purchase orders for reordering
* Called from Reports menu, MReports.prg
CLEAR
USE Master INDEX Master
* - -Make the loop, and display goods below reorder
* - -point, and ask the user how many of each to order.
GO TOP
DO WHILE .NOT. EOF()

* - Find out if on_hand plus on_order quantity is
* - - less than reorder point .
IF (Qty + On_order) <= Reorder
    CLEAR

    * - - - - Show status of item to be reordered
    @ 5,5 SAY "Part number: " + Part_no + " " + P_name
    @ 6,5 SAY "On hand " + STR(Qty,4)
    @ 7,5 SAY "On order " + STR(On_order,4)
    @ 8,5 SAY "Reorder " + STR(Reorder,3)
    @ 9,5 SAY "Unit Cost" + STR(Cost,9,2)

    * Ask user how many to order
    @ 12,5 SAY "Order how many?" GET New_order ;
```

```

        PICT "999"
        REPLACE Order_date WITH T_Date
        READ
    ENDIF
    SKIP
    ENDDO(Continue loop until  end of file)

* - - - - When all orders have been placed
* - - - - Make a temporary file of items to be ordered.
CLEAR
? "Preparing Files ..... please wait"
?
? "(Prepare printer while waiting)"
COPY TO Temp FOR New_order > 0

* - -Update On_order field in Master file with new
* - -orders,then set the New_order field back to zero.
REPLACE ALL On_order WITH On_order + New_order
REPLACE ALL New_order WITH 0

* - - - - Use temp file (which contains new orders)
* - - - - Indexed by vendor
USE Temp
INDEX ON UPPER(Vendor) TO Temp

* - - - - Files ready , inform user
CLEAR
? CHR(7)
WAIT "Ready printer and press any key to print orders"
SET PRINT ON
GO TOP

* - - - - Loop through Temp file
DO WHILE .NOT. EOF()

    * For each vendor , print name and address
    This_loop = Vendor
    Mtotal = 0

```

```
? vendor
? Vendor_add
?
?
? "Please send us the following items...."

* - -For each item to be ordered from this vendor
* - -print quantity,item , and price .
DO WHILE Vendor = this_loop .AND. .NOT. EOF()
    ? New_order , P_name , Cost , New_order * Cost
    Mtotal = New_order * Cost + Mtotal
    SKIP
ENDDO

* - - - When done with this vendor, print total
* - - - Cost,and shipping name and address .
?
? "Total cost :          ", Mtotal
?
? "Mail to : My company"
? "          12 - Tayaran street"
EJECT
ENDDO(Continue for each vendor in Temp file)
* - - - - when done, turn off printer , and return to
* - - - - Reports menu .
SET PRINT OFF
RETURN
```

والجزء الأول من البرنامج يبدأ بالتعريف باسم البرنامج ووظيفته واسم البرنامج الذي قام باستدعائه . ثم يقوم بمسح الشاشة وفتح الملف الرئيسي (Master.dbf) وملف الفهرس الخاص به . وذلك من خلال السطور التالية :

```
CLEAR
USE Master INDEX Master
```

والجزء الثاني من البرنامج يتم من خلاله انشاء حلقة تكرارية (Loop) للبحث خلال الملف الرئيسي (Master.dbf) عن الأصناف التي يقل مجموع الكمية الموجودة منها والكمية تحت الطلب عن حد الطلب لهذه الأصناف . كما يتم عرض بيانات هذه الأصناف . ثم يتم سؤال المستخدم عن الكمية المطلوب صرفها من كل صنف من هذه الأصناف . ويتم ذلك من خلال السطور التالية :

```
GO TOP
DO WHILE .NOT. EOF()
    * - - - Find out if on_hand plus on_order
    * - - - quantity is less than reorder point
    IF (Qty + On_order) <= Reorder
        CLEAR

        * - - - Show status of item to be reordered
        @ 5,5 SAY "Part number:" + Part_no + " "+ ;
            P_name
        @ 6,5 SAY "On hand " + STR(Qty,4)
        @ 7,5 SAY "On order " + STR(On_order,4)
        @ 8,5 SAY "Reorder  " + STR(Reorder,3)
        @ 9,5 SAY "Unit Cost" + STR(Cost,9,2)

        * Ask user how many to order
        @ 12,5 SAY "Order how many?" ;
            GET New_order PICT "999"
        REPLACE Order_date WITH T_Date
        READ

    ENDIF
    SKIP
ENDDO(Continue loop until end of file)
```

ويلاحظ من هذه الأوامر أن الكمية الجديدة التي يتم طلبها تخزن في الحقل (New_order) . كما أن حقل تاريخ الطلب (Order_date) يتم تغييره بتاريخ اليوم الحالي الذي يتم فيه طلب الأصناف . وهو التاريخ الذي يتم ادخاله عند تشغيل نظام المخازن من البداية .

والجزء الثالث من البرنامج يتم من خلاله نسخ جميع بيانات الأصناف المطلوبة في ملف مؤقت (temporary file) يسمى (temp) .

ويتم ذلك من خلال السطور التالية :

CLEAR?

? "Preparing Files please wait"

?

? "(Prepare printer while waiting)"

COPY TO Temp FOR New_order > 0

والجزء الرابع يتم من خلاله تعديل بيانات الملف الرئيسي (Master.dbf) بناء على الكميات الجديدة التي تم طلبها من بعض الأصناف . حيث يتم تعديل بيانات حقل الكمية تحت الطلب (On_order) بإضافة الكمية الجديدة التي تم طلبها الى الكمية السابقة . ثم يتم إعادة حقل الكمية الجديدة (New_order) الى الصفر . ويتم ذلك من خلال السطور التالية :

```
* - -Update On_order field in Master file with new
* - -orders, then set the New_order field back to zero.
REPLACE ALL On_order WITH On_order + New_order
REPLACE ALL New_order WITH 0
```

والجزء الخامس يتم من خلاله فتح الملف الوقت (Temp) . كما يتم فهرسته على حقل البائع (Vendor) . وذلك حتى يمكن طباعة طلب الشراء الخاص بكل بائع على حدة . ويتم ذلك من خلال السطور التالية :

USE Temp

INDEX ON UPPER(Vendor) TO Temp

والجزء السادس يتم من خلاله توجيه المستخدم لتجهيز الطابعة ثم تشغيل الطابعة وإنشاء حلقة تكرارية لطباعة أوامر الشراء لكل بائع (Vendor) على حدة . كما يتم إنشاء حلقة تكرارية أخرى داخلها طباعة الأصناف الخاصة بكل بائع في طلب الشراء الخاص به . ويتم ذلك من خلال السطور التالية :

CLEAR?

CHR(7)

WAIT "Ready printer and press any key to print orders"

SET PRINT ON

GO TOP


```
* - - - - Loop through Temp file
DO WHILE .NOT. EOF()
```

ومن خلال الحلقة التكرارية الأولى ، يتم تخزين اسم البائع في متغير ذاكرة (This_loop) . كما يتم انشاء متغير ذاكرة (Mtotal) لحساب المجموع الكلي لأسعار الأصناف في كل طلب شراء . كما يتم طباعة اسم البائع وعنوانه في بداية طلب الشراء . والسطور التالية توضح ذلك :

```
This_loop = Vendor
Mtotal = 0
? vendor
? Vendor_add
?
?
? "Please send us the following items...."
```

ومن خلال الحلقة التكرارية الثانية يتم عرض بيانات الأصناف الخاصة بكل بائع (Vendor) على حدة ، وذلك بالنسبة لكل بائع يتم تخزين اسمه في المتغير (This_loop) . والسطور التالية توضح هذه الحلقة :

```
DO WHILE Vendor = this_loop .AND. .NOT. EOF()
    ? New_order , P_name , Cost , New_order * Cost
    Mtotal = New_order * Cost + Mtotal
    SKIP
ENDDO
```

وعند الانتهاء من بائع معين ، أي قبل الانتقال الى اسم بائع جديد ، يتم طباعة أمر الشراء . كما يتم طباعة التكلفة الكلية لهذا الطلب والعنوان الذي يتم ارسال الأصناف اليه . ثم يتم نقل ورقة الطباعة (Eject) .

والسطور التالية توضح هذه العملية :

```
? "Total cost : " , Mtotal
?
? "Mail to : My company"
? " 12 - Tayaran street"
EJECT
```

ثم تستمر الحلقة التكرارية الخارجية حتى يتم طباعة باقى أوامر الشراء . وبعد ذلك يتم إعادة الطباعة الى وضعها الأصلي (Off) والعودة الى البرنامج الرئيسى (Master.dbf) ، وذلك من خلال السطور التالية :

```
ENDDO(Continue for each vendor in Temp file)
SET PRINT OFF
RETURN
```

١٤ - ٥ برنامج تعديل الملف الرئيسى

يتم الانتقال الى برنامج تعديل الملف الرئيسى عن طريق اختيار الرقم (3) فى قائمة برنامج تشغيل الملف الرئيسى . حيث يتم تنفيذ البرنامج (Medit.prg). والخطوات الأولية (PSEUDOCODE) لهذا البرنامج تكون كالآتى :

- ١- يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به .
- ٢- يتم تكوين حلقة تكرارية لاجراء التعديل .
- ٣- البحث عن رقم الجزء (Part number) المطلوب تعديل بياناته .
- ٤- عند عدم ادخال رقم الجزء (Part number) ، يتم الرجوع الى القائمة .
- ٥- عند العثور على الجزء المطلوب ، يتم عرض بياناته باستخدام شاشة الادخال (IScreen1) .
- ٦- اذا لم يتم العثور على الجزء المطلوب ، يتم تحذير المستخدم واعطائه الفرصة لادخال رقم جديد .
- ٧- الاستمرار فى التعديل حتى يطلب المستخدم الخروج .
- ٨- عند الانتهاء يتم الرجوع الى القائمة الرئيسية لتشغيل الملف الرئيسى .

والبرنامج الذى يحقق هذه الخطوات الأولية يتم كتابته كالآتى :

```
..... Medit.prg
*   Edit the inventory Master file
*   Called from Master menu , MMenu.prg .
USE Master INDEX Master

* - - - - - Start loop for performing edits
Partnumb = "X"
DO WHILE Partnumb # " "

    * - - - Find out what part number to edit.
```

```

CLEAR
@ 2,1 SAY "Edit Inventory Master File"
@ 2,60 SAY DTOCT(T_Date) + " " + TIME()
@ 3,0 SAY Uline
?
?
?
?
Partnumb = SPACE(5)
@ 15,5 SAY "Edit for what part number?" ;
    GET Partnumb
READ

* - - - - - Try to find that part number
Partnumb = UPPER(Partnumb)
SEEK Partnumb
DO CASE

    * - - - If no part number entered, return to
    * - - - Master menu.
    CASE Partnumb = " "
        CLEAR

    * - - - If part number found , edit using
    * - - - Iscreen1 format file
    CASE FOUND()
        SET FORMAT TO Iscreen1
        READ
        SET FORMAT TO

    * - - - Otherwise warn user, and allow another
    * - - - try
    CASE .NOT. FOUND()
        @ 17,5 SAY "There is no part" + "Partnumb"
        @ 24,5 SAY "Press any key to try again"
        WAIT

ENDCASE
ENDDO(Continue editing untill user requests ...exit)

```

* - - - - Return to Master menu

RETURN

والجزء الأول من البرنامج يبدأ كالمادة بالتعريف باسم البرنامج ووظيفته ثم اسم الملف الذى يقوم باستدعائه . ثم يتم فتح الملف الرئيسى وملف الفهرس الخاص به . وكالمادة يتم تكوين حلقة تكرارية للبحث عن رقم الجزء (Part number) الذى يدخله المستخدم فى المتغير (Partnumb) ويتم ذلك من خلال السطور التالية :

USE Master INDEX Master

Partnumb = "X"

DO WHILE Partnumb # " "

ثم يتم عرض عنوان الشاشة والتاريخ والوقت ثم سؤال المستخدم عن الرقم المطلوب . وذلك من خلال السطور التالية :

CLEAR

@ 2,1 SAY "Edit Inventory Master File"

@ 2,60 SAY DTOCT(T_Date) + " " + TIME()

@ 3,0 SAY Uline

?

?

Partnumb = SPACE(5)

@ 15,5 SAY "Edit for what part number?" GET Partnumb

READ

ثم يتم تحويل رقم الجزء الى حروف كبيرة (Uppercase) والبحث عنه بواسطة الأمر (SEEK) . وذلك كالآتى :

Partnumb = UPPER(Partnumb)

SEEK Partnumb

ثم يبدأ البرنامج بعد ذلك فى اتخاذ القرار بناء على رقم الجزء الذى يدخله المستخدم . فاذا لم يدخل المستخدم أى رقم يتم مسح الشاشة والخروج من الحلقة التكرارية مما يؤدي الى الرجوع الى القائمة الرئيسية لتشغيل الملف الرئيسى . أما اذا أدخل رقما وتم العثور على هذا الرقم (FOUND()) فان البرنامج يعرض بيانات هذا الجزء من خلال شاشة الادخال (IScreen1) التى سبق انشاؤها . ثم يتيح للمستخدم تعديل بيانات هذا الجزء . ثم يتم اغلاق ملف شاشة الادخال

.(IScreen1)

وإذا أدخل المستخدم رقما ، ولكن لم يتم العثور على هذا الرقم فإن البرنامج يحذر المستخدم ويعطيه الفرصة لإدخال رقم آخر .

والسطور التالية توضح هذه العملية :

```
DO CASE
  CASE Partnumb = " "
    CLEAR
  CASE FOUND()
    SET FORMAT TO Iscreen1
    READ
    SET FORMAT TO
  CASE .NOT. FORUND()
    @ 17,5 SAY "There is no part " + Partnumb
    @ 24,5 SAY "Press any key to try again"
    WAIT
ENDCASE
ENDDO(Continue editing until user requests exit)
RETURN
```

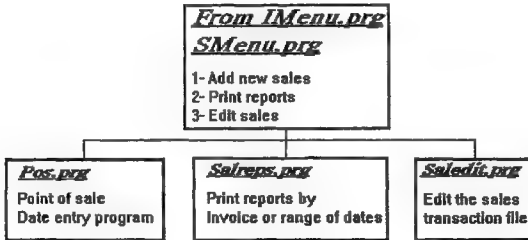

الفصل الخامس عشر

برنامج تشغيل ملف المبيعات

هذا الجزء من برنامج المخازن يختص بمتابعة مبيعات الأصناف وتحديث ملف المبيعات بناء على ذلك . ويتم ذلك عن طريق برنامج مكون من أربعة برامج منفصلة كما سيتم الإيضاح فيما بعد . ويقوم هذا البرنامج بالإشراف على نقطة البيع (Point of Sale) ، حيث يقوم بتجهيز فواتير البيع بناء على رقم الجزء الذي يدخله المستخدم . ويقوم البرنامج كذلك بملء بيانات الأصناف في الفاتورة ألياً . كما يقوم البرنامج بتحذير المستخدم إذا أدخل رقم جزء غير موجود . ويتيح له ادخال رقم جديد . كما يقوم البرنامج بعد ذلك بتجميع أسعار الأجزاء في الفاتورة بالإضافة الى تخزين أرقام الفواتير .

١٥ - ١ تركيب برنامج المبيعات

يتكون هذا البرنامج من أربعة برامج كما سبق الإيضاح . وهي برنامج (SMenu.prg) الذي يتحكم في البرامج الثلاث الأخرى ، وبرنامج (Pos.prg) الذي يتحكم في نقطة البيع (Point of sale) ، وبرنامج (Salreps.prg) الذي يطبع التقارير التي توضح موقف مبيعات الأصناف ، وبرنامج (Saledit.prg) الذي يسمح بتعديل بيانات ملف المبيعات . والتركيب الهرمي للبرنامج يتضح من الشكل التالي :



شكل (١٥ - ١)

وفي الجزء التالي سوف يتم شرح برامج (SMenu) ، (Pos) ، (Salreps) مع تأجيل شرح برنامج (Saledit) الى الفصل الخاص بتحديث الملفات .

١٥ - ٢ برنامج القائمة الرئيسية (smenu.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (2) في القائمة الرئيسية لنظام المخازن . وعند تشغيل البرنامج تظهر القائمة التالية على الشاشة .

Sales System Menu		02/20/90	04:50:30
1 .	Enter point of sale routine		
2 .	Print sales reports		
3 .	Edit sales data		
4 .	Return to main menu		
Enter choice (1 - 4)			

شكل (١٥ - ٢)

والبرنامج لا يختلف تركيبه عن أي برنامج قائمة من البرامج السابقة . ولذلك فليست هناك حاجة إلى كتابة الخطوات الأولية (PSEUDOCODE) له ، ونكتفي فقط بعرض سطور البرنامج كالآتي :

```
***** SMenu.prg
*   Menu of sales portion of the inventory system
*   Called from inventory system Main menu .

* - - - - - Set up loop for presenting menu
Schoice = 0
DO WHILE Schoice # 4
    CLEAR
    @ 2,1 SAY "Sales system menu"
    @ 2,60 SAY DTOC(T_Date) + " " + TIME()
    3,0 SAY Uline
```

```

?
?
TEXT

    1. Enter point of sale routine
    2. Print sales reports
    3. Edit sales data
    4. Return to main menu

ENDTEXT
@ 24,1 SAY "Enter choice (1 - 4) " ;
  GET Schoice PICT "9"
READ

* - - - - - Branch to appropriate choices
DO CASE
    CASE Schoice = 1
        DO Pos
    CASE Schoice = 2
        DO Salreps
    CASE Schoice = 3
        DO Saledit
ENDCASE
ENDDO(While schoice #4)
* - - - - - when done , return to main menu
RETURN

```

١٥ - ٣ برنامج نقطة البيع (Pos.prg)

قبل البدء في شرح أوامر هذا البرنامج سيتم أولاً شرح ما يظهر على الشاشة عند تنفيذه حتى يكون القارئ متتبهاً لوظائف هذا البرنامج . فعند تشغيل البرنامج تظهر شاشة تشبه فاتورة البيع تماماً كما هو واضح من الشكل (١٥ - ٣) .

<u>02/20/90</u>	<u>Invoice number :</u>			
<u>Clerk</u> [REDACTED]	<u>Customer :</u> [REDACTED]			
Part #	Name	Qty	Price	Total

شكل (١٥ - ٢)

والبرنامج يملأ البيانات الموجودة أعلى الفاتورة آلياً مثل التاريخ ورقم الفاتورة كما يقوم المستخدم بكتابة اسم الموظف القائم بعملية البيع (Clerk) واسم العميل الذي يتم البيع له . ثم يظهر عمود ضوئي لأدخال رقم الجزء فيه كما يتضح من الشكل (١٥ - ٤) .

<u>02/20/90</u>	<u>Invoice number :</u>			
<u>Clerk</u> [REDACTED]	<u>Customer :</u> [REDACTED]			
Part #	Name	Qty	Price	Total
[REDACTED]				

شكل (١٥ - ٤)

وعند ادخال المستخدم لرقم جزء غير موجود يظهر التحذير التالي :
No such part !!
ويمكن للمستخدم في هذه الحالة أن يحاول مرة ثانية . وعند كتابة رقم جزء موجود فإن اسم هذا الجزء يظهر ألياً ، كما تظهر أعمدة ضوئية (Highlights) لادخال الكمية (Qty) ، وسعر البيع (Selling Price) . كما يقوم البرنامج بحساب السعر الكلي عن طريق ضرب الكمية (Qty) في سعر البيع (Price) ثم يظهر العمود الضوئي الخاص برقم الجزء التالي حتى يقوم المستخدم بادخال رقم جديد . وتوضح هذه العملية من الشكليين التاليين .

<u>02/20/90</u>		<u>Invoice number :</u>		
<u>Clerk</u> [REDACTED]		<u>Customer :</u> [REDACTED]		
Part #	Name	Qty	Price	Total
A-124	Shoe	30	20	

شكل (١٥ - ٥)

<u>02/20/90</u>		<u>Invoice number :</u>		
<u>Clerk</u> [REDACTED]		<u>Customer :</u> [REDACTED]		
Part #	Name	Qty	Price	Total
A-124	Shoe	30	20	600

شكل (١٥ - ٦)

وعند ادخال المستخدم لرقم جزء جديد موجود في المخزن بالاضافة الى ادخال الكمية والسعر لهذا الصنف ، يقوم المستخدم بتنفيذ نفس العملية السابقة ويتضح ذلك من الشكل التالي :

<u>02/20/90</u>		<u>Invoice number :</u>		
<u>Clerk</u> [REDACTED]		<u>Customer :</u> [REDACTED]		
Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600
A-122	Shirt	20	25	500

شكل (١٥ - ٧)

وعندما يدخل المستخدم رقما موجودا ولكنه ليس الجزء الذي يريده (وهذا يتضح له من اسم الجزء الذي يظهر أليا) فيمكنه في هذه الحالة الضغط على مفتاح (<) مرتين للرجوع الى مكان رقم الجزء وكتابة رقم جزء آخر . كما يمكن الضغط على مفتاح الادخال مرتين لتنفيذ نفس العملية السابقة ، حيث أن الضغط على مفتاح الادخال دون ادخال أي عدد في الكمية (Qty) أو في السعر (Price) يغير البرنامج أن هذا الجزء غير مطلوب ، وبالتالي يعود المؤشر الى مكان رقم الجزء لادخال رقم جزء جديد .

وعندما ينتهي المستخدم من ادخال بيانات هذه الفاتورة (Invoice) ، فإنه يضبط على مفتاح الادخال عندما يكون المؤشر واقفا على العمود الضوئي الخاص برقم الجزء (Part #) . وفي هذه الحالة يقوم البرنامج بتجميع السعر الكلي للفاتورة كما يعرض رسالة للمستخدم لسؤاله اذا كان يريد طباعة هذه الفاتورة أم لا . ويتضح هذا من الشكل التالي :

<u>02/20/90</u>		<u>Invoice number :</u>		
<u>Clerk</u> [REDACTED]		<u>Customer :</u> [REDACTED]		
Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600
A-122	Shirt	20	25	500
Total :				1100
Print Invoice ? (Y/N)				

شكل (١٥ - أ)

وعندما يكتب المستخدم (Y) يتم طباعة هذه الفاتورة (Invoice) ، ثم يقوم البرنامج بمسح الشاشة وعرض سؤال للمستخدم اذا كان يريد طباعة فاتورة أخرى فاذا أجاب بنعم (Y) يتم عرض فاتورة جديدة بنفس الطريقة و برقم فاتورة جديد . واذا أجاب بلا (N) يتم الرجوع الى قائمة البيع .

١٥ - ٢ - ١ كتابة الخطوات الاولى (PSEUDOCODE)

كما سبق الايضاح فان البرنامج الخاص بالتحكم في نقطة البيع (Point of sale) يتم تسميته (Pos.prg) . ويتم كتابة الخطوات الاولى له كالآتي :

- ١ - يتم الحصول على آخر رقم فاتورة من ملف المبيعات (Sales.dbf).
- ٢ - يتم انشاء ملف مؤقت (Temporary File) لمتابعة حركة البيع .
- ٣ - يتم فتح الملف الرئيسي والملف المؤقت .
- ٤ - يتم ربط الملف الرئيسي بالملف المؤقت .

- ٥- يتم تكوين حلقة تكرارية لعرض الفواتير على الشاشة . حيث يتم أولا عرض السطور الأولى من التقرير .
- ٦- يتم تكوين حلقة تكرارية لكل جزء يتم ادخاله الى الفاتورة .
- ٧- يتم تكوين حلقة تكرارية لاختبار رقم الجزء والتأكد من وجوده في الملف الرئيسي (Master.abf) .
- ٨- عند عدم ادخال أى رقم يتم الخروج من البرنامج .
- ٩- عند ادخال رقم غير موجود يتم تحذير المستخدم لادخال رقم جديد .
- ١٠- يتم ادخال الكمية والسعر لكل جزء .
- ١١- عند ادخال كمية (صفر) يتم ادخال رقم جديد .
- ١٢- يتم حساب السعر الكلي للصف وعرضه تحت العنوان (Total) .
- ١٣- يتم تخزين هذه الحركة في ملف مؤقت (Temporary File) .
- ١٤- يتم زيادة اعداد رقم السطر لعرض بيانات الصف التالي .
- ١٥- عند الوصول الى آخر سطر في الشاشة يتم زحزحة الشاشة (Scrolling) سطرا واحدا الى اعلى .
- ١٦- يتم الاستمرار في تنفيذ الحلقة التكرارية حتى يتوقف المستخدم عن ادخال الأصناف . وفي هذه الحالة يتم عرض السعر الكلي للفاتورة (Grand Total) ثم يتم عرض سؤال للمستخدم عما اذا كان يريد طباعة الفاتورة أم لا .
- ١٧- يتم سؤال المستخدم اذا كان يريد طباعة فاتورة أخرى ، وبناء على ذلك تستمر الحلقة التكرارية حتى يطلب المستخدم الخروج .
- ١٨- يتم اغلاق ملفات قواعد البيانات (Database Files) كما يتم تحديث ملف المبيعات (Sales File) .
- ١٩- يتم مسح الملف المؤقت (Temporary File) .
- ٢٠- يتم الرجوع الى القائمة الرئيسية للمبيعات .

١٥ - ٢ - ٢ كتابة البرنامج

قبل كتابة البرنامج يجب ملاحظة أن هذا البرنامج طويل بعض الشيء . وربما يحتاج الى برنامج معالج كلمات آخر غير البرنامج الموجود مع برنامج (DBase III +) وذلك في حالة كتابته متضمنا كل سطور الملاحظات . ولذلك يفضل عدم كتابة سطور الملاحظات خلال البرنامج اذا تم استخدام المصحح الخطي الخاص ببرنامج (DBase III+) . ولكن سنضيف سطور الملاحظات هنا للتوضيح فقط .

ملاحظة

عند الانتهاء من جميع البرامج الخاصة بنظام المخازن يمكن مسح جميع سطور الملاحظات وذلك بعد نسخ البرامج الأصلية في قرص آخر . حيث أن

ذلك يؤدي الى سرعة تنفيذ البرنامج بدرجة كبيرة .

ويتم كتابة سطور البرنامج كالآتي :

```
***** Pos.prg
*   Point of sale data entry program for sales
*   Called from Sales menu , SMenu.prg

* - Get last-used invoice number from the Sales file.
USE Sales
GO BOTT
Minvoice = Invoice_no

* - - - - - Add new transactions to a temporary file
SET SAFETY OFF
COPY STRUCTURE TO Tempinv

* - - - - - Open Master and temporary files
SELECT A
USE Master INDEX Master
SELECT B
USE Tempinv
SET RELATION TO part_no INTO Master
* - - - - - Set pointer in temporary file
StartTrans = 1

* - - - - - Set up loop for displaying invoice forms
Again = "Y"
DO WHILE Again = "Y"
    * - - Set up top portion of invoice on the screen
    CLEAR
    Minvoice = Minvoice + 1
    STORE SPACE (30) TO MClerk, MCust
    Mtotal = 0
    @ 1,2 SAY T_Date
    @ 1,30 SAY "Invoice number:" + STR(Minvoice,5)
    @ 2,2 SAY "Clerk" GET MClerk
```

```

@ 2,35 SAY "Customer:" GET MCust
@ 3,0 SAY Uline
? "PART #      Name", SPACE(20)
?? "Qty      Price      Total"
READ
* - - - Set up loop for each item in the invoice
Row = 7
Adding = .T.
DO WHILE Adding
    Partnumb = SPACE(5)
    ok = .F.
    * - - - loop to check validity of part number
    DO WHILE .NOT. ok
        * - - - Set up invoice memory variables.
        Quantity = 0.00
        Sel_Price = 0.00
        * - - - - - Ask for part number.
        Partnumb = SPACE(5)
        @ Row,2 GET Partnumb
        READ
        * - - - - - Make sure Part number exists.
        partnumb = UPPER(TRIM(Partnumb))
        SELECT A
        SEEK Partnumb

        * - - - - - Decide next step based on
        * - - - - - existance of part number.
        DO CASE
            * - - - - - No part number entered
            CASE LEN(partnumb) = 0
                ok = .T.
                Adding = .F.
            * - - - - - Part number does not exist.
            CASE .NOT. FOUND()
                @ Row, 10 SAY "No such part !!"
                ok = .F.
            * - - - - - Part number exists
            CASE FOUND()

```

```

* - - - - -Display Part name, get
* - - - - - quantity and price
@ Row,10 SAY P_name
@ Row,35 GET Quantity PICT "999.99"
@ Row,40 GET Sel_Price PICT "999.99"
READ
* - - - If quantity is zero , loop
* - - - Else , Display total.
IF Quantity = 0
    LOOP
ELSE
    @ Row,50 SAY Quantity * Sel_Price;
    PICT "###,###.##"
    Mtotal = Mtotal + Quantity * ;
    Sel_price
    ok = .T.
ENDIF

* - - - Add a blank record to the
* - - - Tempinv file , and fill in
* - - - the fields.
SELECT B
APPEND BLANK
REPLACE Date WITH T_Date
REPLACE Clerk WITH M_Clerk
REPLACE Invoice_no WITH Minvoice
REPLACE Customer WITH MCust
REPLACE Part_no WITH Partnumb
REPLACE Qty WITH Quantity
REPLACE Price WITH Sel_Price
REPLACE Posted WITH .F.

ENDCASE
ENDDO(continue loop for checking part number)
Row = Row + 1
* - - - Scroll screen if reached end of screen.
IF Row >= 19
    @ 24,1
    ?

```

```

        Row = 19
    ENDIF
    ENDDO(while still adding items to invoice)
    * - - - Display grand total, and pause before next
    * - - - - invoice.
    @ Row+2,40 SAY "Total:"
    @ Row+2,50 SAY Mtotal PICT "##,###.##"
    Pinvoice = "Y"
    @ 23,2 SAY "Print invoice ? (Y/N)" GET Pinvoice ;
    PICT "!"
    READ
    * - - - - Print invoice , Reset Start Trans.
    IF Pinvoice = "Y"
        SET PRINT ON
        ? "Date:" , T_Date
        ? "Invoice number:" , STR(Minvoice,5)
        ? "Customer:" , MCust , SPACE(20)
        ?? "Clerk:" , Mclerk
        ? Uline
        ?
        SELECT B
        GOTO StartTrans
        LIST OFF WHILE .NOT. EOF() Part_no, ;
        A --> P_name, Qty , Price, Qty * Price
        ?
        ?
        ? "Total:" , SPACE(34), Mtotal
        EJECT
        SET PRINT OFF
        StartTrans = RECCOUNT() + 1
    ENDIF
    CLEAR
    @ 23,2 SAY "Do another transaction? (Y/N)" ;
    GET Again PICT "!"
    READ
    ENDDO(add invoices while user does not request exit)

```

```
* - - - - - Close databases and update sales file.
CLOSE DATABASES
CLEAR
? "Updating transaction file , please wait ..."
SET TALK ON
USE SALES
APPEND FROM Tempinv
USE Tempinv
ZAP
USE Sales INDEX Sales
REINDEX
SET TALK OFF
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يبدأ كالمادة بالتعريف باسم البرنامج ووظيفته
تسم اسم البرنامج الذي قام باستدعائه . ثم يتم فتح ملف المبيعات (Sales.dbf)
دون استخدام أى فهرس معه . وذلك للحصول على آخر رقم فاتورة (Invoice_no) . حيث أن رقم الفاتورة يتوقف على ترتيب ادخال هذه الفاتورة في الملف . ثم يتم تخزين هذا الرقم في المتغير (Minvoice) . ويتم ذلك من خلال السطور التالية :

```
USE Sales
GO BOTT
Minvoice = Invoice_no
```

والجزء الثاني يوضح استخدام الملف المؤقت (Tempinv.dbf) في تخزين الفواتير الجديدة قبل نقلها الى ملف المبيعات (Sales.dbf) .
وتوفر هذه الطريقة سرعة كبيرة لتنفيذ البرنامج . وبعد الانتهاء من ادخال الفواتير المطلوبة ، يقوم البرنامج باضافة هذه الفواتير الى ملف المبيعات .
وبلاحظ هنا استخدام الأمر (SET SAFETY OFF) حتى يتم نسخ هيكل ملف المبيعات في الملف المؤقت دون ظهور رسالة التحذير المعتادة في هذه الحالة والسطور التالية توضح هذه العملية .

```
SET SAFETY OFF
COPY STRUCTURE TO Tempinv
```

والجزء الثالث يتم من خلاله فتح الملف المؤقت والملف الرئيسي في مناطق عمل مختلفة . ثم يتم ربط الملفين بناء على حقول رقم الجزء (Part number) . وهذا يساعد على اختبار رقم الجزء الذي يدخله المستخدم والتأكد من وجوده في الملف الرئيسي . كما أن الربط بين الملفين يساعد بعد ذلك على الحصول على اسم الجزء حتى يتم كتابته في الفاتورة . والسطور التالية توضح فتح هذين الملفين وإنشاء العلاقة (Relation) بينهما .

```
SELECT A
USE Master INDEX Master
SELECT B
USE Tempinv
SET RELATION TO part_no INTO Master
```

وعند طباعة الفواتير يجب أن يعرف البرنامج رقم السجل الذي يبدأ بطابعته . لذلك يتم إنشاء متغير الذاكرة (StartTrans) وأعطائه القيمة (1) من خلال السطر التالي .

```
StartTrans = 1
```

والجزء الرابع يتم من خلاله تكوين حلقة تكرارية لإضافة الفواتير (Invoices) ويتم استخدام المتغير (Again) في التحكم في هذه الحلقة كما يتم من خلال الحلقة زيادة قيمة المتغير (Minvoice) بواحد عند الانتقال إلى فاتورة جديدة .

كما يتم إنشاء المتغير (MClerk) ، والمتغير (MCust) لكل فاتورة جديدة لتسجيل اسم الموظف القائم بالبيع واسم العميل في كل فاتورة . ثم يتم عرض السطور العلوية للتقرير . وذلك من خلال السطور التالية :

```
* - - - - Set up loop for displaying invoice forms
Again = "y"
DO WHILE Again = "y"
    * - - Set up top portion of invoice on the screen
    CLEAR
    Minvoice = Minvoice + 1
    STORE SPACE (30) TO MClerk, MCust
    Mtotal = 0
    @ 1,2 SAY T_Date
```

```
@ 1,30 SAY "Invoice number:" + STR(Minvoice,5)
@ 2,2 SAY "Clerk" GET MClerk
@ 2,35 SAY "Customer:" GET MCust
@ 3,0 SAY Uline
? "PART #    name", SPACE(20)
?? "Qty      Price      Total"
READ
```

والجزء الخامس يتم من خلاله تكوين حلقة تكرارية لادخال الاصناف في الفاتورة . ويتم انشاء المتغير (Adding) للتحكم في هذه الحلقة . ويبدأ عرض الاصناف ابتداء من السطر رقم (7) على الشاشة . وتستمر الحلقة التكرارية حتى يضغط المستخدم على مفتاح الادخال دون ادخال رقم جزء (Part number) . والسطور التالية توضح هذه العملية .

```
Row = 7
Adding = .T.
DO WHILE Adding
```

والجزء السادس يتم من خلاله تكوين حلقة تكرارية أخرى لاختيار رقم الجزء الذي يدخله المستخدم . وذلك بعد انشاء متغير الذاكرة (Partnumb) . وكذلك انشاء المتغير المنطقي (ok) للتحكم في الحلقة التكرارية . ويتم اعطاء هذا للمتغير القيمة (.F.) حتى يتم تنفيذ الحلقة التكرارية مرة واحدة على الأقل . والسطور التالية توضح هذه العملية .

```
Partnumb = SPACE(5)
ok = .F.
DO WHILE .NOT. ok
```

كما يتم انشاء متغير ذاكرة (Quantity) لتخزين الكمية المباعة من الصنف، والمتغير (Sel_Price) لتخزين سعر هذا الصنف . وذلك من خلال السطور التالية .

```
Quantity = 0.00
Sel_Price = 0.00
```

واستخدام نقطة العلامة العشرية هنا مهم لتخزين الأعداد متضمنة رقمين عشريين .

والجزء السابع يتم من خلاله سؤال المستخدم عن رقم الجزء المطلوب ادخاله . ويتم تخزين هذا الرقم في المتغير (Partnumb) ، ثم يتم تحويل هذا الرقم الى حروف كبيرة (Uppercase) حتى يتم البحث عنه بواسطة الأمر (SEEK) . والسطور التالية توضح هذه العملية .

```
Partnumb = SPACE(5)
@ Row,2 GET Partnumb
READ
Partnumb = UPPER(TRIM(Partnumb))
SELECT A
SEEK Partnumb
```

والجزء الثامن يتم من خلاله اتخاذ القرار بناء على ما يدخله المستخدم في المتغير (Partnumb) . فعندما يضغط المستخدم على مفتاح الادخال دون ادخال أى رقم جزء يتم تخزين القيمة (.T.) في المتغير (ok) والقيمة (.F.) في المتغير (Adding) وذلك للخروج من الحلقة التكرارية . حيث أن ذلك معناه أن المستخدم يريد الخروج . والسطور التالية توضح هذه العملية .

```
DO CASE
CASE LEN(partnumb) = 0
ok = .T.
Adding = .F.
```

وعندما يتم ادخال رقم جزء غير موجود في الملف يتم تحذير المستخدم واعطاؤه الفرصة للمحاولة مرة ثانية . ويتم ذلك عن طريق تخزين القيمة (.F.) في المتغير (ok) حتى يتم تنفيذ الحلقة التكرارية مرة ثانية . ويتم ذلك من خلال السطور التالية :

```
CASE .NOT. FOUND()
@ Row, 10 SAY "No such part !!"
ok = .F.
```

وعندما يتم ادخال رقم موجود في الملف ، فإن البرنامج يأتى بباقي البيانات الخاصة بهذا الصنف مثل اسم الصنف (P_name) ثم يسأل عن

الكمية المطلوبة من هذا الصنف وسعر البيع . ويتم ذلك من خلال السطور التالية :

```
CASE FOUND()
* - - - - Display Part name, get
* - - - - quantity and price
@ Row,10 SAY P_name
@ Row,35 GET Quantity PICT "999.99"
@ Row,40 GET Sel_Price PICT "999.99"
READ
```

والجزء التاسع يتم من خلاله حساب السعر لكل صنف ثم حساب السعر الكلي للفاتورة . كما يتم من خلال هذا الجزء أيضا إعادة تنفيذ الحلقة التكرارية باستخدام الأمر (LOOP) في حالة ادخال المستخدم للكمية (صفر) . وذلك عندما يضغط على مفتاح الادخال عند وقوف المؤشر على العمود الضوئي (Highlight) الخاص بالكمية . ويتم ذلك من خلال السطور التالية :

```
IF Quantity = 0
  LOOP
ELSE
  @ Row,50 SAY Quantity * Sel_Price;
  PICT "##,###.##"
  Mtotal = Mtotal + Quantity * Sel_Price
  ok = .T.
ENDIF
```

والجزء العاشر يتم من خلاله نقل البيانات التي تم ادخالها في متغيرات الذاكرة الى الحقول المقابلة في الملف الوقت (Tempinv) . وذلك من خلال السطور التالية :

```
SELECT B
APPEND BLANK
REPLACE Date WITH T Date
REPLACE Clerk WITH M_Clerk
REPLACE Invoice_no WITH Minvoice
REPLACE Customer WITH M_Cust
REPLACE Part_no WITH Partnumb
REPLACE Qty WITH Quantity
```

```
REPLACE Price WITH Sel_Price
REPLACE Posted WITH .P.
ENDCASE
ENDDO(continue loop for checking part number)
```

والجزء التالي يتم من خلاله زيادة عداد السطور بواحد لكتابة بيانات الصنف التالي . وعند الوصول إلى نهاية الشاشة يتم زحزحة الشاشة (Scrolling) بمقدار سطر واحد لأعلى . ويتم ذلك عن طريق تحريك المؤشر إلى آخر سطر على الشاشة دون كتابة أى شيء فيه ، وذلك مع تثبيت رقم السطر (Row) عند الرقم (١٩) . ويتم ذلك من خلال السطور التالية :

```
Row = Row + 1
* - - - - Scroll screen if reached end of screen.
IF Row >= 19
    @ 24,1
    ?
    Row = 19
ENDIF
ENDDO(while still adding items to invoice)
```

وعند الانتهاء من إدخال الأصناف في هذه الفاتورة ، يعرض البرنامج السعر الكلى (Total) لهذه الفاتورة ويسأل المستخدم إذا كان يريد طباعة الفاتورة أم لا . والسطور التالية توضح هذه العملية :

```
@ Row+2,40 SAY "Total:"
@ Row+2,50 SAY Mtotal PICT "##,###.##"
Pinvoice = "Y"
@ 23,2 SAY "Print invoice ? (Y/N)" ;
GET Pinvoice PICT "!"
READ
```

وعندما يكتب المستخدم (Y) يتم تشغيل الطابعة . ثم يتم كتابة رأس التقرير (Heading) . ثم يتم تحريك مؤشر السجلات (Record pointer) إلى أول سجل في الملف المؤقت (Tempinv) . وهو السجل الذى تم تخزين رقمه فى المتغير (StartTrans) . ويتم عرض بيانات الصنف فى الفاتورة وحساب السعر الكلى لهذا الصنف ثم

الانتقال الى الصنف التالي وعرض بياناته وهكذا حتى نهاية الملف . وعند الانتهاء من ادخال كل الأصناف في الفاتورة يتم تقديم صفحة جديدة على الطابعة باستخدام الأمر (EJECT) للتجهيز لطباعة فاتورة جديدة عندما يريد المستخدم ذلك . ويتم تخزين الرقم الممثل لعدد السجلات في الملف المؤقت (Tempinv) زائداً واحد في متغير الذاكرة (StartTrans) . وذلك لكي تبدأ الفاتورة التالية من هذا الرقم عندما يريد المستخدم ارسال فاتورة أخرى باسم بائع آخر (Vendor) . والسطور التالية توضح هذه العملية .

```
IF Pinvoice "Y"
  SET PRINT ON
  ? "Date:" , T_Date
  ? "Invoice number:" , STR(Minvoice,5)
  ? "Customer:" , MCust , SPACE(20)
  ?? "Clerk:" , MClerk
  ? Uline
  ?
  SELECT B
  GOTO StartTrans
  LIST OFF WHILE .NOT. EOF() Part_no, A --> P_name,;
  Qty , Price, Qty * Price
  ?
  ?
  ? "Total:" , SPACE(34), Mtotal
  EJECT
  SET PRINT OFF
  StartTrans = RECCOUNT() + 1
ENDIF
```

وفي الجزء التالي يتم سؤال المستخدم اذا كان يريد طباعة فواتير أخرى . فاذا أراد ذلك يتم تنفيذ الحلقة مرة ثانية حيث يكون التغير (Again) مخزناً فيه القيمة ("Y") . أما اذا أراد الخروج فإنه يكتب (N) . وفي هذه الحالة يتوقف تنفيذ الحلقة . ويتم ذلك من خلال السطور التالية :

```
CLEAR
@ 23,2 SAY "Do another transaction? (Y/N)" ;
GET Again PICT "!"
```

```
READ
ENDDO(add invoices while user does not request exit)
```

وفي الجزء التالي يقوم البرنامج بإضافة السجلات الموجودة في الملف المؤقت (Tempinv) إلى آخر ملف المبيعات ، كما يتم تحديث ملف الفهرس . ويقوم البرنامج أيضا بمسح السجلات الموجودة في الملف المؤقت لتوفير المساحة التخزينية على القرص . ثم يقوم البرنامج بإغلاق جميع الملفات المفتوحة والعودة إلى البرنامج الرئيسي لتشغيل برنامج المبيعات (SMenu.prg) . ويتم ذلك من خلال السطور التالية :

```
CLOSE DATABASES
CLEAR
? "Updating transaction file , please wait ..."
SET TALK ON
USE SALES
APPEND FROM Tempinv
USE Tempinv
ZAP
USE Sales INDEX Sales
REINDEX
SET TALK OFF
CLOSE DATABASES
RETURN
```

١٥ - ٣ - ٢ إدخال السعر آليا

يجدر الإشارة هنا إلى أن هذا البرنامج عام ، يمكن استخدامه في أي نقطة بيع . لذلك فإنه يبدو طويلا ومعقدا بعض الشيء . كما يمكن تعديله ليناسب الظروف والمطالب المختلفة للمستخدم . حيث يمكن مثلا إدخال السعر آليا في فاتورة البيع (Invoice) بدلا من كتابته بواسطة المستخدم . ولأجراء هذا التعديل يلزم أولا إضافة حقل سعر الجزء (Price) في الملف الرئيسي (Master.dbf) ثم تعديل البرنامج بناء على ذلك .

١٥ - ٤ برنامج تقارير البيع

يتم تنفيذ هذا البرنامج عندما يختار المستخدم الرقم (2) من قائمة تشغيل ملف المبيعات (SMenu) . وهذا البرنامج يتيح للمستخدم عرض أو طباعة تقارير أما لفاتورة محددة أو لحركة المبيعات خلال فترة محددة (أي من تاريخ محدد الى تاريخ آخر) . وعند تشغيل هذا البرنامج تظهر الشاشة التالية :

Sales Report Options	02/20/90	09:30:25
1. By invoice number 2. By dates 3. Return to sales menu		
Enter choice (1-3)		

شكل (١٥ - ٩)

وعندما يختار المستخدم الرقم (١) من قائمة برنامج تقارير مبيعات الأصناف لعرض أو طباعة تقرير عن فاتورة محددة يظهر السؤال التالي :
 Look for what invoice number?
 وعندما يقوم المستخدم بإدخال رقم الفاتورة (Invoice number) يظهر التقرير الخاص بهذه الفاتورة على الشاشة أو على الطابعة كالآتي مثلا :

Invoice number : 1246		Date : 02/20/90		
Clerk : Hassan		Customer: D. PATHY		
Part #	Qty	Part name	Price	Total
AAA	1	printer	900	900
BBB	10	floppy disk	16	160

شكل (١٥ - ١٠)

وعندما يختار المستخدم الرقم (2) من القائمة لعرض بيانات حركة البيع خلال فترة محددة ، يظهر الآتي على الشاشة :

Enter start date : / /

Enter end date : / /

وعندما يكتب المستخدم تاريخ البداية وتاريخ النهاية يظهر التقرير الخاص بحركة المبيعات خلال هذه الفترة على الشاشة أو على الطابعة كالاتي :

Page No. 1 02/20/90							
<i>Sales Transactions</i>							
Date	Invoice	Sales person	Customer	Part	Qty	Sale	Price
02/01/90	1243	Magdy	Salem	SAF	20	700	
02/10/90	1368	Medhat	Shawky	LAM	10	500	
** TOTAL **				1200			

شكل (١٥ - ١١)

ويمكن اداء التقرير باستخدام الأمر (CREATE) أو الأمر (MODIFY) وذلك كالاتي :

CREATE REPORT Sales

على أن تكون محتويات الأعمدة كالاتي :

Column	Contents	Heading	Width	Decimals	Total?
1	Date	Date	9		
2	Invoice_no	Invoice	7		
3	Clerk	Salesperson	14		
4	Customer	Customer	14		
5	Part_no	Part	6		
6	Qty	Qty	5	0	N
7	Cost	Sale Price	10	0	Y

شكل (١٥ - ١٢)

١٥ - ٤ - ١ كتابة الخطوات الأولية للبرنامج

كما سبق الايضاح فان الوظيفة الرئيسية لبرنامج تقارير المبيعات (Salreps.prg) هي سؤال المستخدم عن نوع التقرير الذى يريده . ثم عرض هذا التقرير على الشاشة أو طباعته على الطابعة . والخطوات الأولية لهذا البرنامج كالآتي :

- ١ - يتم فتح ملف المبيعات (Sales.dbf) والملف الرئيسى (Master.dbf) .
- ٢ - يتم ربط الملفين .
- ٣ - يتم تكوين حلقة تكرارية لعرض قائمة الاختيارات .
- ٤ - يتم سؤال المستخدم عن نوع التقرير المطلوب .
- ٥ - في حالة طلب التقرير بناء على الفاتورة يتم سؤال المستخدم عن رقم الفاتورة المطلوبة .
- ٦ - يتم البحث عن أول سجل يحتوى على هذا الرقم فى ملف المبيعات (Sales.dbf) .
- ٧ - عند العثور على هذا السجل يتم طباعة رأس التقرير بناء على البيانات الموجودة فى هذا السجل .
- ٨ - يتم طباعة بيانات الأصناف الخاصة برقم الفاتورة المطلوبة .
- ٩ - فى حالة طلب التقارير خلال فترة زمنية محددة يتم سؤال المستخدم عن تاريخ البداية وتاريخ النهاية .
- ١٠ - يتم تحويل التواريخ التى يدخلها المستخدم من الصورة الحرفية الى الصورة التاريخية .
- ١١ - يتم طباعة التقارير خلال هذه الفترة باستخدام صورة التقرير (Sales.frm) .
- ١٢ - عند الانتهاء يتم الرجوع الى القائمة الرئيسية لبرنامج المبيعات (SMenu.prg) .

١٥ - ٤ - ٢ كتابة البرنامج

يتكون برنامج تقارير البيع من السطور التالية :

```
***** SalReps.prg
*      Print reports from the sales file
*      Called from sales menu, SMenu.prg
```

```

* - - - - - Open files and set up relationship.
SELECT 1
USE Sales
SELECT 2
USE Master INDEX Master

SELECT 1
SET RELATION TO Part_no INTO Master
Repchoice = 0

* - - - - - Start loop for menu
DO WHILE Repchoice # 3
    CLEAR
    @ 2,1 SAY "Sales Report Options"
    @ 2,60 SAY DTOC(T_Date) + " " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT
        1. By invoice number
        2. By dates
        3. Return to Sales menu
    ENDTEXT
    @ 24,1 SAY "Enter choice(1-3)" ;
    GET Repchoice PICT "9" RANGE 1,3
    READ

* - - - - - IF not exiting , ask about printer.
@ 5,0 CLEAR
STORE " " TO YN, Printer
IF Repchoice # 3
    @ 15,5 SAY "Send to printer " GET YN PICT "!"
    READ
    * - - - - - Set up printer macro
    IF YN = "Y"
        Printer = "TO PRINT"
    ENDIF
ENDIF
ENDIF

* - - - - -Print appropriate report based on request.

```

```

@ 5,0 CLEAR
DO CASE
    * - - - - Case 1 : Search by invoice number.
    CASE Repchoice = 1
        @ 15,5
        INPUT "Look for what invoice?" TO Isearch
        CLEAR
        LOCATE FOR invoice_no = Isearch
        * - - - If found , print invoice.
        IF FOUND()
            IF YN = "Y"
                SET PRINT ON
            ENDIF
            * - - Print header from first record
            * - - with that invoice number.
            ? "Invoice number : " , Invoice_no
            ?? "Date:" , Date
            ? "Clerk:" , Clerk,"Customer:" , Customer
            ?
            ? "Part # Part name Qty Price Total "

            * - - - Print data for all records with
            * - - - that invoice number.
            LIST OFF WHILE Invoice_no = Isearch ;
                Part_no,B-->P_name, Qty , Price , ;
                (Qty * Price)
            IF YN = "Y"
                EJECT
                SET PRINT OFF
            ENDIF
        ENDIF(found)

    * - - - - Case 2 : Search by dates
    CASE Repchoice = 2
        STORE SPACE(8) TO Start , End
        @ 15,5 SAY "Enter start date" GET Start ;
        PICT " 99/99/99"
        @ 17,5 SAY "Enter end date" ;

```

```

GET End PICT "99/99/99"
READ
Start = CTOD(Start)
End = CTOD(End)
CLEAR
* - - - - - Print the report.
REPORT FORM Sales FOR Date >= Start .AND.;
      Date <= End & Printer
ENDCASE

* - - - IF not going to printer, Pause the screen
IF YN # "Y" .AND. Repchoice # 3
  ?
  ?
  WAIT "Press any key to return to the reports ;
      menu"
ENDIF
ENDDO(while user does not request to exit)

* - - - - - When done return to sales menu
SET RELATION TO
CLOSE DATABASES
RETURN

```

والجزء الأول من البرنامج يبدأ كالمعتاد باسم البرنامج (Salreps.prg) ووظيفته والبرنامج القائم باستدعائه ، ثم أوامر فتح الملفات . ويلاحظ هنا أن ملف المبيعات (Sales.dbf) يتم فتحه بدون فتح الفهرس معه . وذلك لأن المطلوب طباعة تقارير تعتمد على رقم الفاتورة (Invoice number) أو التاريخ (Date) . وحيث أن البيانات يتم تخزينها فعليا في ملف المبيعات مرتبة حسب رقم الفاتورة وتاريخ ادخالها ، لذلك فإن الملف يكون مرتبا بالترتيب المطلوب دون الحاجة الى استخدام الفهرس . ولادخال اسم الصنف في الفاتورة يلزم فتح الملف الرئيسي (Master.dbf) وربطه بملف المبيعات . والسطور التالية توضح هذه الخطوات .

```

* * * * * SalReps.prg
*   Called from sales menu , SMenu.prg
* - - - - - Open files and set up relationship.

```

```

SELECT 1
USE Sales
SELECT 2
USE Master INDEX Master
SELECT 1
SET RELATION TO Part_no INTO Master

```

والجزء الثاني يتم من خلاله تكوين حلقة تكرارية لعرض قائمة الاختيارات على المستخدم وسؤاله عن الاختيار المطلوب ثم تخزين هذا الاختيار في متغير الذاكرة (Repchoice) . والسطور التالية توضح هذه الخطوات :

```

Repchoice = 0
* - - - - - Start loop for menu
DO WHILE Repchoice # 3
    CLEAR
    @ 2,1 SAY "Sales Report Options"
    @ 2,60 SAY DTOC(T_Date) + " " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT

        1. By invoice number
        2. By dates
        3. Return to Sales menu

    ENDTXT
    @ 24,1 SAY "Enter choice(1-3)" ;
    GET Repchoice PICT "9" RANGE 1,3
    READ

```

والجزء الثالث يوضح انشاء متغير الذاكرة (Printer) وإدخال القيمة (TO PRINT) في هذا المتغير لاستخدامه كماكرو لتشغيل الطابعة بعد ذلك ، وذلك في حالة عدم اختيار المستخدم للرقم (3) من القائمة للخروج . ويتم ذلك من خلال السطور التالية :

```
* - - - -IF not exiting , ask about printer.
@ 5,0 CLEAR
STORE " " TO YN, Printer
IF Repchoice # 3
    @ 15,5 SAY "Send to printer " GET YN PICT "!"
    READ
    * - - - - Set up printer macro
    IF YN = "Y"
        Printer = "TO PRINT"
    ENDIF
ENDIF
ENDIF
```

والجزء الرابع يوضح استخدام الأمر (DO CASE) في طباعة التقرير المطلوب حسب اختيار المستخدم . فإذا أراد المستخدم الطباعة بناء على رقم الفاتورة ، فإن البرنامج يسأل عن رقم الفاتورة المطلوب كما يحدد أول سجل يحتوى على هذا الرقم . ومن هذا السجل يقوم البرنامج بعرض عنوان التقرير (Heading) الذي يحتوى على اسم الموظف القائم بعملية البيع (Clerk) واسم العميل المشتري (Customer) وتاريخ البيع . ثم يستخدم الأمر (LIST) والعبارة (WHILE) لمرس أو طباعة بيانات جميع السجلات التي تشترك في رقم الفاتورة . والسطور التالية توضح هذه العملية :

```
* - - - -Print appropriate report based on request.
@ 5,0 CLEAR
DO CASE

    * - - - - Case 1 : Search by invoice number.
    CASE Repchoice = 1
        @ 15,5
        INPUT "Look for what invoice ? " TO Isearch
        CLEAR
        LOCATE FOR invoice_no = Isearch
        * - - - - If found , print invoice.
        IF FOUND()
            IF YN = "Y"
                SET PRINT ON
            ENDIF
            * - - - - Print header from first record
```

```

* - - - - with that invoice number.
? "Invoice number : " , Invoice_no
?? "Date:" , Date
? "Clerk:" , Clerk, "Customer:" , Customer
?
? "Part # Part name Qty Price Total "
* - - - Print data for all records with that
* - - - - invoice number.
LIST OFF WHILE Invoice_no = Isearch , ;
      Part_no,B-->P_name, Qty , Price , ;
      (Qty * Price)
IF YN = "Y"
      EJECT
      SET PRINT OFF
ENDIF
ENDIF(found)

```

والجزء الخامس يتم من خلاله عرض بيانات التقرير في حالة طلب المستخدم تقريراً بالبيانات التي تمت خلال فترة معينة . وفي هذه الحالة يسأل البرنامج عن تاريخ البداية (Start) وتاريخ النهاية (End) ثم يستخدم صورة التقرير (Sales.frm) التي سبق أنشاؤها . ويتم ذلك من خلال الأسطر التالية :

```

* - - - - Case 2 : Search by dates
CASE Repchoice = 2
      STORE SPACE(8) TO Start , End
      @ 15,5 SAY "Enter start date" GET Start ;
      PICT " 99/99/99"
      @ 17,5 SAY "Enter end date" ;
      GET End PICT "99/99/99"
      READ
      Start = CTOD(Start)
      End = CTOD(End)
      CLEAR
      * - - - - Print the report.
      REPORT FORM Sales FOR Date >= Start .AND. ;
      Date <= End & Printer
ENDCASE

```

وبلاحظ هنا عملية تحويل التاريخ من الصورة الحرفية التي يدخلها المستخدم الى الصورة التاريخية التي يستطيع البرنامج التعامل معها ، وذلك باستخدام الدالة (CTOD) .

والجزء السادس يتم من خلاله إيقاف الشاشة لحظيا (Pause) في حالة عدم الرغبة في طباعة التقرير والاكتفاء بعرضه على الشاشة . ويتم ذلك من خلال السطور التالية :

```
* - - - -IF not going to printer, Pause the screen
IF YN # "Y" .AND. Repchoice # 3
?
?
WAIT "Press any key to return to the reports ;
menu"
ENDIF
```

والجزء السابع يتم من خلاله إنهاء الحلقة التكرارية وإغلاق الملفات المفتوحة ثم العودة الى البرنامج القائم بالاستدعاء (SMenu.prg) . ويتم ذلك من خلال السطور التالية :

```
ENDDO(while user does not request to exit)
* - - - - When done return to sales menu
SET RELATION TO
CLOSE DATABASES
RETURN
```

الفصل السادس عشر

برنامج تشغيل ملف الاضافة

هذا البرنامج هو جزء من برنامج المخازن (Inventory) يختص بتسجيل بيانات الاصناف التي يتم توريدها الى المخزن . ويتم تخزين بيانات هذه الاصناف في ملف الاضافة (Newstock.dbf) الذي سبق انشاؤه . ويتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (3) من القائمة الرئيسية لبرنامج المخازن . انظر الشكل (١ - ١٦) .

<u>Inventory system main menu</u>	<u>02/20/90</u>
<ol style="list-style-type: none"> 1. Manage master inventory 2. Record sales 3. Record new stock 4. Exit 	
Enter choice 0	

شكل (١ - ١٦)

<u>New stock system menu</u>	<u>02/20/90</u>	<u>08:30:45</u>
<ol style="list-style-type: none"> 1. Record new items 2. Print new stock data 3. Edit new stock data 4. Return to main menu 		
Enter choice (1 - 4)		

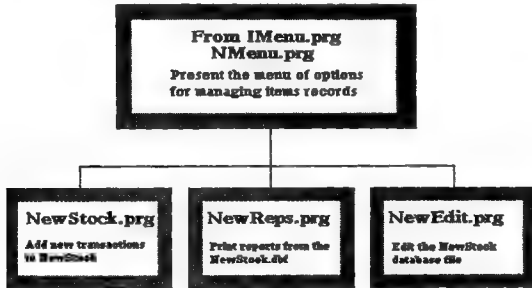
شكل (٢ - ١٦)

وفي هذه الحالة تظهر القائمة الخاصة بالاضافة . انظر الشكل (١٦ - ٢) .
وهذه القائمة تتيح للمستخدم اضافة حركة جديدة (Transaction) ، أو طباعة
تقارير ، أو تصحيح بيانات الحركة ، أو الرجوع الى القائمة الرئيسية للمخازن .

١٦ - ١ تركيب البرنامج

يتكون برنامج الاضافة من أربعة برامج منفصلة . البرنامج الأول هو برنامج
(NMenu.prg) الذي يقوم بالتحكم في تشغيل البرامج الثلاثة الأخرى . وبناء على
اختيار المستخدم يتم التفرع الى البرنامج (NewStock.prg) الذي يسمح للمستخدم
باضافة بيانات الأصناف الجديدة . أو يتم التفرع الى البرنامج (NewRep.prg)
الذي يطبع التقارير . أو يتم التفرع الى البرنامج (NewEdit.prg) الذي يسمح
بتعديل بيانات أى حركة اضافة .

والشكل التالى يوضح التركيب الهرمى للبرنامج .



شكل (١٦ - ٢)

١٦ - ٢ برنامج قائمة الاضافة (NMenu.prg)

وهذا البرنامج لا يختلف عن برامج القوائم التي سبق شرحها لذلك يتم عرض سطور البرنامج دون الحاجة الى شرحها مرة ثانية وهي كالآتي :

```
***** NMenu.prg
* Menuf or managing New Stock portion of Inventory
* system . Called from Inventory System main menu.
* - - - - - Set up loop form presenting menu.
Nchoice = 0
DO WHILE Nchoice # 4
    CLEAR
    @ 2,1 SAY "New Stock System menu"
    @ 2,60 SAY DTC(T_Date) + " " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT
        1. Record new items
        2. Print new stock reports
        3. Edit new stock data
        4. Return to main menu
    ENDTEXT
    @ 24,1 SAY "Enter choice(1-4)"
    GET Nchoice PICT "9" RANGE 1,4
    READ

    * - - - - - Branch to appropriate program.
DO CASE
    CASE Nchoice = 1
        DO NewStock
    CASE Nchoice = 2
        DO NewReps
    CASE Nchoice = 3
        DO NewEdit
ENDCASE
```

```
ENDDO(while Nchoice # 4)
* - - - - - Return to main menu
RETURN
```

١٦ - ٢ برنامج ادخال بيانات الاصناف (Newstock.prg)

عندما يختار المستخدم الرقم (1) من قائمة الاضافة فان برنامج القائمة (NMenu.prg) ينتقل الى البرنامج (NewStock.prg) لينفذه . وفي هذه الحالة يتم مسح الشاشة وتظهر الرسالة التالية للمستخدم :

```
Enter data for goods received
Part number :-
```

وعندما يكتب المستخدم رقما غير موجود في قاعدة البيانات يتم تحذير المستخدم بصفارة (Beep) ثم تظهر رسالة توضح له عدم وجود هذا الجزء ، مع اعطائه الفرصة للمحاولة مرة ثانية . وذلك كالآتي :

```
Enter data for goods received
Part number :-                :No such part !!!
```

ويجب ملاحظة أن هذا البرنامج لا يضيف اصناف بأرقام جديدة ، ولكنه يضيف كميات من اصناف موجودة أرقامها في قاعدة البيانات . وهذا عكس البرنامج (Addnums.prg) المستخدم في برنامج تشغيل الملف الرئيسي (Master.dbf) .

وعند ادخال المستخدم لرقم جزء موجود يقوم البرنامج بعرض اسم هذا الجزء على الشاشة بالاضافة الى تاريخ ادخال هذا الصنف (تاريخ اليوم الحالي) واسم البائع (Vendor) . كما يتيح له تعديل اسم البائع (Vendor) حسب الحاجة ثم يطلب من المستخدم كتابة الكمية الواردة من هذا الصنف وسعر الشراء . وذلك كالآتي :

```
Enter data for goods received
Part number  A-122           Shirt
Quantity : -                Price:
Date 02/20/90              Vendor : Hasan
```

وعند الانتهاء من ادخال البيانات في الحقول الخالية ، فان البرنامج يسأل عن الصنف التالي المطلوب اضافته . وعند الانتهاء من ادخال الاصناف ، يقوم المستخدم بالضغط على مفتاح الادخال للرجوع الى قائمة الاضافة مرة ثانية .

١٦ - ٤ كتابة الخطوات الاولى (PSEUDOCODE)

يتم كتابة الخطوات الاولى للبرنامج كالآتي :

- ١ - يتم فتح الملف الرئيسي (Master.dbf) وملف الاضافة (NewStock.dbf).
- ٢ - يتم تكوين حلقة تكرارية لتسجيل بيانات الاصناف الجديدة .
- ٢ - عند ادخال رقم الجزء يقوم البرنامج بالبحث عن هذا الجزء في الملف الرئيسي (Master.dbf).
- ٤ - عند عدم العثور على رقم الجزء يتم تحذير المستخدم والسماح له بادخال رقم جديد .
- ٥ - عند العثور على رقم الجزء يتم عرض بياناته وسؤال المستخدم عن الكمية الواردة وسعر الشراء .
- ٦ - يتم الاستمرار في ادخال الاصناف حتى يطلب المستخدم الخروج .
- ٧ - يتم العودة الى قائمة الاضافة .

١٦ - ٥ كتابة البرنامج

يتم كتابة هذا البرنامج كالآتي :

```

..... NewStock.prg
*   Data entry program for goods received
*   Called from New Stock, NMenu.prg
SELECT A
USE Master INDEX Master
SELECT B
USE NewStock INDEX NewStock
SELECT A

* - - - - Set up loop for recording goods received.
Partnumb = "x"
DO WHILE Partnumb # " "
    @ 5,0 CLEAR
    
```

```

Partnumb = SPACE(5)
@ 10,2 SAY "Enter data for goods received"
@ 12,4 SAY "Partnumber" GET Partnumb
READ
* - If a part number was entered , find it in
* - Master file
IF Partnumb # " "
    SEEK Partnumb
    DO CASE

        * - - - If part not found, warn user and
        * - - - - try again
        CASE .NOT. FOUND()
            @ 12,23 SAY "No such part !!!"
            ? CHR(7)
            * - If found Append a new record to
            * - NewStock.dbf, and get
            * - rest of data.
            CASE FOUND()
                @ 12,25 SAY P_name
                SELECT B
                APPEND BLANK
                REPLACE Part_no WITH Partnumb
                REPLACE Date WITH T_Date
                REPLACE Vendor WITH A -> Vendor
                @ 14,2 SAY "Quantity" GET Qty
                @ 14,22 SAY "Price" GET COST PICT ;
                    "99999.99"
                @ 16,2 SAY "Date" GET Date PICT ;
                    "99/99/99"
                @ 16,22 SAY "Vendor" GET Vendor
                READ
                SELECT A
            ENDCASE
        ENDIF(Partnumb# " ")
    ENDDO(While user does not want to quit)

```

```
* - - - - - Close files and return to New Stock
* - - - - - menu.
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يقوم بفتح الملف الرئيسي (Master.dbf) واختبار رقم الجزء الذي يتم ادخاله . كما يتم فتح ملف الأمانة (NewStock.dbf) لتخزين بيانات الأصناف الجديدة . ويتم ذلك من خلال السطور التالية :

```
SELECT A
USE Master INDEX Master
SELECT B
USE NewStock INDEX NewStock
SELECT A
```

والجزء الثاني يقوم بتكوين حلقة تكرارية لادخال رقم الجزء . وذلك من خلال السطور التالية :

```
Partnumb = "x"
DO WHILE Partnumb # " "
    @ 5,0 CLEAR
    Partnumb = SPACE(5)
    @ 10,2 SAY "Enter data for goods received"
    @ 12,4 SAY "Partnumber" GET Partnumb
    READ
```

والجزء الثالث يتم من خلاله البحث عن الرقم الذي قام المستخدم بادخاله وذلك من خلال السطور التالية :

```
IF Partnumb # " "
    SEEK Partnumb
```

والجزء الرابع يتم من خلاله تحذير المستخدم في حالة ادخال رقم جزء غير موجود . وذلك من خلال السطور التالية :

```
DO CASE
    CASE .NOT. FOUND()
        @ 12,23 SAY "No such part !!!"
        ? CHR(7)
```

والجزء الخامس يتم من خلاله عرض اسم الصنف الذي يتم ادخال رقمه على الشاشة ، وذلك في حالة ادخال المستخدم لرقم موجود . كما يتم اضافة سجل خال في نهاية ملف الاضافة (NewStock.dbf) حتى يتم ادخال البيانات الجديدة فيه . كما يتم ملء بيانات رقم الصنف والتاريخ واسم البائع آليا وسؤال المستخدم عن الكمية والسعر واسم البائع في حالة تغييره . ثم يتم التجهيز لادخال صنف جديد . ويتم ذلك من خلال السطور التالية :

```
CASE FOUND()
    @ 12,25 SAY P_name
    SELECT B
    APPEND BLANK
    REPLACE Part_no WITH Partnumb
    REPLACE Date WITH T_Date
    REPLACE Vendor WITH A -> Vendor
    @ 14,2 SAY "Quantity" GET Qty
    @ 14,22 SAY "Price" GET COST PICT "99999.99"
    @ 16,2 SAY "Date" GET Date PICT "99/99/99"
    @ 16,22 SAY "Vendor" GET Vendor
    READ
    SELECT A
```

والجزء السادس يتم من خلاله اغلاق الملفات بعد انتهاء الحلقة التكرارية ثم العودة الى قائمة الاضافة . وذلك من خلال السطور التالية :

```
ENDCASE
ENDIF(Partnumb# " ")
ENDDO(While user does not ask to quit)
CLOSE DATABASES
RETURN
```


١٦ - ٦ برنامج تقارير الاضافة (NewReps.prg)

يسمح هذا البرنامج للمستخدم بمراجعة بيانات الأصناف الجديدة في صورة تقارير بناء على رقم الجزء (Part number) أو خلال فترة زمنية محددة . وتفيد هذه التقارير في حل أي خلافات قد تنشأ مع البائعين (Vendors) عن طريق إمدادهم بالبيانات الدقيقة عن الأصناف وأسعارها .

ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (2) من قائمة الاضافة .
انظر الشكل (١٦ - ٤) .

New Stock System Menu 02/20/90 02:20:45
1. Record new items
2. Print new stock data
3. Edit new stock data
4. Return to main
Enter choice (1 - 4) 0

شكل (١٦ - ٤)

في هذه الحالة تظهر القائمة التالية على الشاشة .

1. By part number
2. By dates
3. Return to New Stock menu

وعند اختيار المستخدم للرقم (1) يظهر سؤال آخر عما إذا كان المطلوب طباعة التقرير أم عرضه فقط على الشاشة . ثم يظهر سؤال آخر عن رقم الجزء المطلوب عرض أو طباعة التقرير له . وعند كتابة المستخدم لهذا الرقم يظهر تقرير يوضح حركة الأضافة لهذا الصنف .

وعند اختيار المستخدم للرقم (2) من القائمة ، فإن هذا يعني أنه يريد عرض حركة الأصناف من تاريخ معين إلى تاريخ آخر . وفي هذه الحالة يظهر على الشاشة سؤال عن تاريخ البداية وتاريخ النهاية كالآتي :

Enter start date : / /

Enter end date : / /

فيقوم المستخدم بكتابة المطلوب . وفي هذه الحالة تظهر كل بيانات حركة الأضافة التي تمت بين هذين التاريخين .

أنظر الشكل (١٦ - ٥) ، الشكل (١٦ - ٦)

Page No. 1		02/20/90			
Inventory Items Received					
Part Name	Part	Qty	Purchase Price	Date	Vendor Name
A-122	Shirt	20	500	02/20/90	Hasan
A-122	Shirt	35	750	02/01/90	Salem

الشكل (١٦ - ٥)

Page No. 1			02/20/90		
Inventory Items Received					
Part	Part Name	Qty	Purchase Price	Date	Vendor Name
A-122	Shirt	30	750	02/01/90	Hasan
A-122	Shirt	20	500	02/20/90	Salem
A-121	Shoss	20	500	02/25/90	Hytham

شكل (١٦ - ٦)

ولكتابة البرنامج يتم أولاً إنشاء صورة التقرير المطلوب . مع ملاحظة أنه يمكن إنشاء صورة واحدة للتقرير واستخدامها في حالة طلب التقرير بناء على رقم الجزء أو بناء على فترة زمنية محددة . ويتم ذلك باستخدام الأمر (CREATE) أو الأمر (MODIFY) . وحيث أن التقرير يجب أن يتضمن بيانات من ملف الاضافة (NewStock.dbf) ، وكذلك بيانات من الملف الرئيسي (Master.dbf) ، لذلك يلزم عند إنشاء التقرير فتح ملف الاضافة وكذلك الملف الرئيسي والربط بينهما . ويتم ذلك من خلال الأوامر التالية :

```
CLEAR ALL
SELECT 1
USE NewStock
SELECT 2
USE Master INDEX Master
SELECT 1
SET RELATION TO Part_no INTO Master
MODIFY REPORT NewStock
```

مع ملاحظة أن هذه الأوامر تكتب من مشيرة النقطة (Dot Prompt) قبل بداية كتابة البرنامج .

وعندما تظهر الشاشة الخاصة بإنشاء التقرير يتم ادخال محتويات أعمدة التقرير (Columns) كما يظهر من الشكل التالي :

Column	Contents	Heading	Width	Decimals	Total
1	Part_no	Part	6		
2	B->P_name	Part Name	15		
3	Qty	Qty	4	2	N
4	Cost	Purchase price	12	2	N
5	Date	Date	8		
6	Vendor	Vendor	25		

شكل (١٦ - ٧)

ويلاحظ هنا كتابة (B ->P_name) للحصول على اسم الجزء من الملف الرئيسي (Master.dbf) .

١٦ - ٧ كتابة البرنامج

نظرا لأن هذا البرنامج شبيه ببرامج التقارير التي سبق كتابتها مع الملف الرئيسي (Master.dbf) وملف المبيعات (Sales.dbf) ، فسوف يكتفى هنا بكتابة البرنامج فقط ويمكن الرجوع الى برامج التقارير السابقة لمتابعة شرحها . والبرنامج يتكون من السطور التالية :

```
*****NewReps.prg
*   Print reports from the NewStock file
*   Called from NewStock menu, NMenu.prg .

* - - - - - Open NewStock and Master databases.
SELECT 1
USE NewStock
```

```

SELECT 2
USE Master INDEX Master
* - - - - - Set up relationship.
SELECT 1
SET RELATION TO Part_no INTO Master

* - - - - - Start loop for menu
Repchoice = 0
DO WHILE Repchoice # 3
    CLEAR
    @ 2,1 SAY "New Stock Report options"
    @ 2,60 SAY DTOC(T_Date) + " " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT
        1. By part number
        2. By dates
        3. Return to New Stock menu
    ENDTXT
    @ 24,1 SAY "Enter choice(1-3)" ;
    GET Repchoice PICT "9"
    READ

    * - - - - - If not exiting , ask about printer.
    @ 3,0 CLEAR
    STORE " " TO YN, Printer
    IF Repchoice # 3
        @ 15,5 SAY "Send to printer?" GET YN PICT "!"
        READ
        * - - - - - Set up printer macro.
        IF YN = "Y"
            Printer = "TO PRINT"
        ENDIF
    ENDIF
ENDIF

```

```
* - - - - - If not going to pritner, pause the
* - - - - - screen.
IF YN # "Y" .AND. Repchoice # 3
  ?
  ?
  WAIT "Press any key to return to the Repchoice;
      menu"
ENDIF
ENDDO (while user does not request to exit)
* - - - - - when done , return to New Stock menu
SET RELATION TO
CLOSE DATABASES
RETURN
```

الفصل السابع عشر

برنامج تحديث البيانات

في الأبواب السابقة تم انشاء ثلاثة أجزاء رئيسية من نظام المخازن للتحكم في الملف الرئيسي (Master.dbf) وملف للمبيعات (Sales.dbf) وملف للاضافة . (NewStock.dbf) وفي هذا الباب يتم انشاء البرنامج المكمل للنظام الذي يمثل أهم جزء فيه . وهذا البرنامج يقوم بتحديث (Updating) لبيانات الملف الرئيسي (Master.dbf) بناء على البيانات الموجودة في ملف المبيعات (Sales.dbf) وملف للاضافة (NewStock.dbf) . وفي هذا الباب أيضا يتم انشاء برنامج تعديل ملف المبيعات وبرنامج تعديل ملف الاضافة .

١٧ - ١ برنامج تحديث الملف الرئيسي (Master.dbf)

يقوم هذا البرنامج بخصم الكميات الأصناف الموجودة في ملف المبيعات من الكميات الموجودة في الملف الرئيسي لكل صنف تم البيع منه . كما يقوم باضافة كميات الأصناف الموجودة في ملف الاضافة الى الكميات الموجودة في الملف الرئيسي .

ويقوم هذا البرنامج أيضا بخصم الكميات الموجودة في ملف الاضافة من الكميات الموجودة في حقل تحت الطلب (On_order) في الملف الرئيسي ، حيث أن هذه الأصناف تم اضافتها بالفعل . كما يقوم بتعديل سعر الشراء لهذه الأصناف (Purchase Price) بالسعر الموجود في ملف الاضافة ، حيث أن هذا السعر يعتبر أحدث سعر للصنف . وأخيرا يقوم البرنامج بتعديل تاريخ آخر تحديث للبيانات بالتاريخ الموجود في ملف الحركة المستخدم .

و يتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (4) من القائمة الرئيسية لبرنامج تشغيل الملف الرئيسي (Master.dbf) . انظر الشكل (١٧ - ١)

Manage Master Inventory	02/20/90	12:40:50
<ol style="list-style-type: none"> 1. Add new part numbers 2. Print reports 3. Make changes 4. Update from sales and Newstock 5. Return to main menu 		
Enter choice (1 - 5)		

شكل (١٧ - ١)

ولكتابة هذا البرنامج يتم أولا كتابة الخطوات الأولية (PSEUDOCODE) .

١٧ - ١ - ١ كتابة الخطوات الأولية (PSEUDOCODE)

- ١ - يتم مسح الشاشة .
- ٢ - يتم عرض رسالة توضح للمستخدم أن الملف الرئيسي جارى تحديثه من ملف المبيعات .
- ٣ - يتم فتح ملف المبيعات والفهرس الخاص به .
- ٤ - يتم نسخ جميع سجلات ملف المبيعات التي لم يتم تحديثها (لم يتم ترحيل بياناتها) الى ملف مؤقت (Temporary file) .
- ٥ - يتم فتح الملف المؤقت .
- ٦ - يتم التأكد من وجود سجلات في الملف المؤقت (التأكد أن الملف المؤقت ليس فارغا) .
- ٧ - في حالة وجود سجلات في الملف المؤقت ، يتم فتح الملف الرئيسي وملف الفهرس الخاص به .
- ٨ - يتم تحديث الملف الرئيسي من الملف المؤقت وذلك بخصم كميات الأصناف المباعة وتعديل حقل تاريخ آخر تحديث .
- ٩ - يتم اغلاق جميع الملفات .
- ١٠ - يتم فتح ملف المبيعات (Sales.dbf) .
- ١١ - يتم ادخال القيمة (T.) أى صحيح في حقل الترحيل (Posted) لجميع سجلات ملف المبيعات . لأن هذا يوضح أن جميع السجلات قد تم ترحيلها (Posted) ..
- ١٢ - يتم اغلاق جميع الملفات .
- ١٣ - يتم عرض رسالة للمستخدم أن الملف الرئيسي جارى تحديثه من ملف الاضافة (Newstock.dbf) .
- ١٤ - يتم فتح ملف الاضافة وملف الفهرس الخاص به .
- ١٥ - يتم نسخ جميع سجلات ملف الاضافة التي لم يتم تحديثها (لم يتم ترحيل بياناتها) الى ملف مؤقت (Temporary file) .
- ١٦ - يتم فتح الملف المؤقت .
- ١٧ - يتم التأكد من وجود سجلات في الملف المؤقت .
- ١٨ - في حالة وجود سجلات في الملف المؤقت ، يتم فتح الملف الرئيسي وملف الفهرس الخاص به .
- ١٩ - يتم تحديث الملف الرئيسي من الملف المؤقت وذلك باضافة كميات الأصناف المضافة وتعديل سعر الصنف وتاريخ آخر تحديث له وكذلك طرح كمية الصنف المضافة على الخزن من الكمية الموجودة في حقل تحت الطلب (On_order) .

- ٢٠- يتم اغلاق جميع الملفات .
- ٢١- يتم فتح ملف الاضافة (NewStock.dbf) .
- ٢٢- يتم ادخال القيمة (.T.) أى صحيح فى حقل الترحيل (Posted) لجميع سجلات ملف الاضافة . حيث أن هذا يوضح أن جميع السجلات تم ترحيلها . و ذلك حتى لايعاد ترحيلها مرة ثانية .
- ٢٣- يتم اغلاق جميع الملفات .
- ٢٤- يتم الرجوع الى قائمة برنامج تشغيل الملف الرئيسى (MMenu.prg) .

١٧ - ١ - ٢ كتابة البرنامج

هذا البرنامج يتم تسميته (Updater.prg) ويتكون من السطور التالية :

```
***** Updater.prg
*   Update the Master File from Sales and NewStock.
*   Called from Master menu , MMenu.prg.

* - - - - - Ask user if sure about updating.
YesNo = "Y"
@ 5,0 CLEAR
@ 15,4 SAY "Update Master file from Sales and" + ;
        "NewStock?(Y/N)" GET YesNo PICT "!"
READ
IF YesNo = "N"
    RETURN
ENDIF

* - - - - - Display a message that Master is being
* - - - - - updated from Sales file.
@ 5,0 CLEAR
@ 15,5 SAY "Updating from the Sales file ..."
USE Sales INDEX Sales

* - - - - - Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

```
* - - - - Make sure there are records in Temp
SELECT 2
USE Temp
IF RECCOUNT()>0
    * Use Master file and index for updating
    SELECT 1
    USE Master INDEX Master

    * - - - - - Update from the temporary sales file.
    UPDATE ON Part_no FROM Temp REPLACE Qty WITH ;
        Qty - Temp -> Qty , Date WITH Temp -> Date

    * - - - - - USE original Sales file , make all
    * - - - - - posted fields "True".
    CLOSE DATABASES
    USE Sales
    REPLACE ALL Posted WITH .T.
ENDIF(record count >0)
CLOSE DATABASES

* - - - - - Display a message that Master is bieng
* - - - - - updated from the NewStock file.
@ 15,5 SAY "Updating from the NewStock file"
USE NewStock INDEX NewStock
* - - - - - Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted

* - - - - - Make sure there are records in temp.
SELECT 2
USE Temp
IF RECCOUNT() > 0
    * - - - - - USE Master file and index for updating
    SELECT 1
    USE Master INDEX Master
    * - - - - - Update from the temporary NewStock file.
    UPDATE ON Part_no FROM Temp ;
    REPLACE Qty WITH Qty + Temp -> Qty , Date WITH ;
```

```
Temp -> Date, Cost WITH Temp -> Cost, ;
On_order WITH On_order - Temp -> Qty

* - - USE original NewStock file, make all posted
* - - - - - fields "True"
CLOSE DATABASES
USE NewStock
REPLACE ALL Posted WITH .T.
ENDIF(record count > 0)

* - - - free up all work areas , and return to Master
* - - - menu.
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يبدأ بالتعريف باسم البرنامج (Updater.prg) ووظيفته ثم اسم البرنامج الذي قام باستدعائه (MMenu.prg). ثم يقوم بمسح الشاشة وعرض رسالة للمستخدم للتأكد من رغبته في التحديث (Updating). ثم يتم عرض رسالة توضح للمستخدم أن الملف الرئيسي (Master.dbf) جاري تحديثه من ملف المبيعات. ثم يقوم البرنامج بفتح ملف المبيعات (Sales.dbf) وملف الفهرس الخاص به. ثم يقوم بنسخ السجلات التي لم يتم ترحيلها إلى ملف مؤقت (Temp.dbf). ويتم تنفيذ هذه الخطوات من خلال السطور التالية :

```
YesNo = "Y"
@ 5,0 CLEAR
@ 15,4 SAY "Update Master file from Sales and " + ;
"NewStock?(Y/N)" GET YesNo PICT "!"
READ
IF YesNo = "N"
RETURN
ENDIF
* - - - - - Display a message that Master is being
* - - - - - updated from Sales file.
@ 5,0 CLEAR
@ 15,5 SAY "Updating from the Sales file ..."
USE Sales INDEX Sales
```

```
* - - - - - Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

والجزء الثاني من البرنامج يقوم بفتح الملف المؤقت للمبيعات (Temp.dbf) الذي يحتوي على كل السجلات المطلوب ترحيل بياناتها الى الملف الرئيسي . ثم يقوم بالتأكد من وجود سجلات في هذا الملف وذلك لتجنب التعامل مع ملف قاعدة بيانات فارغ ، لأن ذلك يمكن أن يؤدي الى نتائج غير متوقعة . ثم يتم فتح الملف الرئيسي (Master.dbf) والفهرس الخاص به في منطقة عمل أخرى . ويتم ذلك من خلال السطور التالية :

```
* Make sure there are records in Temp file.
SELECT 2
USE Temp
IF RECCOUNT(>)>0
    * Use Master file and index for updating
    SELECT 1
    USE Master INDEX Master
```

والجزء الثالث من البرنامج يتم من خلاله اجراء عمليات الترحيل بخصم الكميات الموجودة في الملف المؤقت (Qty -> Temp) من الكميات الموجودة في الملف الرئيسي (Qty) ثم يتم تعديل تاريخ آخر تحديث للملف الرئيسي (Date) بالتاريخ الموجود في ملف المبيعات . ويتم ذلك من خلال السطور التالية :

```
* - - - - - Update from the temporary sales file.
UPDATE ON Part_no FROM Temp REPLACE Qty WITH ;
Qty - Temp -> Qty , Date WITH Temp -> Date
```

والجزء الرابع يتم من خلاله تمييز السجلات التي تم ترحيلها حتى لا يتم ترحيلها مرة ثانية . ويستخدم حقل الترحيل (Posted) في هذه العملية . حيث يتم ادخال القيمة (.T.) أى (True) صحيح في هذا الحقل لجميع سجلات ملف المبيعات . وهذا يعنى أن جميع البيانات قد تم ترحيلها الى الملف الرئيسي . والسطور التالية توضح هذه العملية .

```
Close DATABASES
USE Sales
```

REPLACE ALL Posted WITH .T.

ثم يتم اغلاق جميع الملفات حتى يمكن البدء في تحديث الملف الرئيسي من ملف الاضافة . ويتم ذلك من خلال السطور التالية :

```
ENDIF(record count > 0)
CLOSE DATABASES
```

والجزء الخامس يتم من خلاله تنفيذ عملية التحديث من خلال ملف الاضافة (NewStock.dbf) بنفس الطريقة السابقة ، حيث يتم أولاً عرض رسالة على المستخدم توضح له اجراء عملية التحديث ثم يتم فتح ملف الاضافة والفهرس الخاص به . ثم يتم نسخ جميع السجلات التي لم يتم تحديثها (ترحيلها من ملف الاضافة) الى ملف مؤقت (Temp) . وحيث أنه قد سبق كتابة الأمر (SET SAFETY OFF) فسي البرنامج الرئيسي لبرنامج المخازن (IMenu.prg) ، فان عملية نسخ السجلات تتم دون ظهور الرسالة التحذيرية المعتادة في حالة النسخ فوق ملف سابق . والسطور التالية توضح تنفيذ هذه الخطوات .

```
@ 15,5 SAY "Updating from the NewStock file"
USE NewStock INDEX NewStock
* - - - - - Copy unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

ثم يتم التأكد من وجود سجلات في الملف المؤقت (Temp) . وذلك من خلال السطور التالية :

```
SELECT 2
USE Temp
IF RECCOUNT() > 0
```

ثم يتم فتح الملف الرئيسي (Master.dbf) وملف الفهرس الخاص به من خلال السطور التالية :

```
SELECT 1
USE Master INDEX Master
```

ثم يقوم البرنامج بعد ذلك بترحيل القيم الموجودة في حقول الملف المؤقت (Temp) الى الملف الرئيسي (Master) . وذلك باضافة الكمية الموجودة في حقل الكمية (Qty -> Temp) على الكمية الموجودة في الملف الرئيسي (Qty) . كما يتم استبدال السعر بالسعر الموجود في الملف المؤقت (Temp->cost) وكذلك تاريخ آخر تحديث . كما يتم طرح الكمية (Temp->Qty) من الكمية الموجودة في حقل تحت الطلب (On_order) . والسطور التالية توضح تنفيذ هذه الخطوات .

```
* - - - - - Update from the temporary NewStock file.
UPDATE ON Part_no FROM Temp ;
REPLACE Qty WITH Qty + Temp -> Qty , Date WITH ;
Temp -> Date, Cost WITH Temp -> Cost, ;
On_order WITH On_order - Temp -> Qty
```

وفي الجزء الأخير من البرنامج يتم ادخال القيمة (.T.) في كل حقول الترحيل (Posted) حتى لا يتم ترحيلها مرة ثانية . ثم يتم اغلاق جميع الملفات والعودة الى قائمة تشغيل الملف الرئيسي (MMenu.prg) . والسطور التالية توضح هذه الخطوات :

```
* - - - USE original NewStock file, make all
* - - - posted fields "True"
CLOSE DATABASES
USE NewStock
REPLACE ALL Posted WITH .T.
ENDIF(record count > 0)

* - - - - - free up all work areas , and return to
* - - - - - Master menu.
CLOSE DATABASES
RETURN
```

١٧ - ٢ برنامج تصحيح ملف المبيعات (SalEdit.prg)

كما يلاحظ من عرض البرامج الفرعية المختلفة المكونة لبرنامج المخازن ، فقد تم تاجيل شرح البرنامج الخاص بتصحيح ملف المبيعات (SalEdit.prg) الى هذا

الفصل . والسبب في ذلك أن التصحيح بعد عملية الترحيل يؤدي الى عدم مطابقة بيانات الملف الرئيسي (Master.dbf) للواقع . فمثلا لو فرضنا أن المستخدم باع عشرة طابعات ولكنه كتب في حقل الكمية الخاص بهذا الصنف عشرين بدلا من عشرة . فاذا قام بتعديل هذا الصنف في ملف المبيعات بعد ترحيله الى الملف الرئيسي (Master) فان هذا التعديل لا يظهر في الملف الرئيسي . وبالتالي لا يصبح الملف الرئيسي معبرا عن الكمية الصحيحة في المخزن . ولذلك فان برنامج تصحيح ملف المبيعات يسمح للمستخدم باجراء أى تعديلات يريد لها ملف المبيعات وفي نفس الوقت يقوم أليا بإدخال هذه التعديلات على الملف الرئيسي . والخطوات المنطقية لتنفيذ هذه العملية تتلخص في الآتي :

- ١- السماح للمستخدم بتعديل أى بيانات سجل معين .
- ٢- إذا قام المستخدم بتعديل بيان رقم جزء (Part number) في ملف المبيعات . يقوم البرنامج بإضافة الكمية المباعة من هذا الصنف على رقم الجزء السابق وطرحها من رقم الجزء الجديد في الملف الرئيسي .
- ٣- إذا قام المستخدم بتعديل الكمية المباعة في ملف المبيعات يتم حساب الفرق بين الكمية التي سبق إدخالها والكمية الجديدة . ثم يتم طرح هذه الكمية من الكمية الموجودة في الملف الرئيسي .

وليس هناك حاجة لتعديل تاريخ آخر تحديث إذا قام المستخدم بتعديله في ملف المبيعات لأنه في جميع الأحوال يمثل تاريخ التحديث (Updating) وليس التعديل (Editing) .

وهذا البرنامج عند تشغيله يؤدي الى عرض الرسالة التالية على الشاشة .

Enter Invoice number to edit (0 if none) :

وعندما يكتب المستخدم رقم الفاتورة المطلوب التعديل فيها ، فان البرنامج يعرض كل الحركة الخاصة بهذه الفاتورة متضمنة بيانات الأصناف المختلفة . وفي هذه الحالة يقوم المستخدم باختيار أحد هذه الأصناف لتعديل بياناته . فتظهر شاشة خاصة بالتعديل متضمنة بيانات هذا الصنف لتعديلها أو مسحها . انظر الشكل (١٧ - ٢) .

ويستطيع المستخدم تحريك المؤشر الى أى بيان خاص بهذا الصنف وتعديله . كما يمكنه كتابة (Y) أمام السؤال الذي يظهر أسفل الشاشة كالآتي :

Delete this record ? (Y/N)

وذلك عندما يريد الغاء هذا الصنف .

ويقوم البرنامج بعد ذلك بإجراء التعديلات المطلوبة على الملف الرئيسي .

Edit Sales Transactions 02/20/90 10:50:30	
Invoice Number :	Date :
Part Number :	
Clerk :	Customer :
Quantity :	Selling Price:
Delete this record (Y/N)	

شكل (١٧ - ٢)

١٧ - ٢ - ١ الخطوات الأولية (PSEUDOCODE)

يتم كتابة الخطوات الأولية للبرنامج كالآتي :

- ١ - يتم إنشاء متغيرات الذاكرة .
- ٢ - يتم تكوين حلقة تكرارية لإدخال رقم الفاتورة .
- ٣ - يتم فتح ملف المبيعات (Sales.dbf) وملف الفهرس الخاص به (Sales.ndx) .
- ٤ - يتم حساب عدد السجلات التي تشترك في رقم الفاتورة الذي يدخله المستخدم .
- ٥ - إذا لم يتم العثور على أى سجل بنفس رقم الفاتورة يتم تحذير المستخدم والسماح له بالمحاولة مرة ثانية مع رقم فاتورة آخر .
- ٦ - إذا كانت هناك عدة سجلات تشترك في رقم الفاتورة . يتم عرض بياناتها للمستخدم وسؤاله عن رقم السجل المطلوب تعديل بياناته . وإذا كان هناك سجل واحد يتم الذهاب إليه .

- ٧- يتم الذهاب الى السجل المطلوب تعديل بياناته في ملف المبيعات وتخزين القيم الموجودة فيه في متغيرات ذاكرة .
- ٨- يتم عرض بيانات السجل من خلال شاشة تصحيح والسماح للمستخدم بتصحيح أى بيانات في السجل .
- ٩- بعد انتهاء عملية التصحيح يتم ضبط بيانات الملف الرئيسى .
- ١٠- اذا كان السجل قد تم ترحيله (Posted) ثم قام المستخدم بمسحه يتم تجهيزه للمسح من الملف الرئيسى .
- ١١- اذا كان السجل قد تم ترحيله وقام المستخدم بتعديل رقم الجزء (Part number) ، يتم الذهاب الى رقم الجزء القديم فى الملف الرئيسى (Master) وإضافة الكمية التى سبق انقاصها منه . كما يتم الذهاب الى رقم الجزء الجديد فى الملف الرئيسى وطرح هذه الكمية منه .
- ١٢- اذا كان السجل قد تم ترحيله (Posted) وقام المستخدم بتعديل الكمية، يتم حساب الفرق بين الكمية السابقة والكمية الجديدة .
- ١٣- يتم الذهاب الى رقم الجزء فى الملف الرئيسى وطرح الفرق من الكمية الموجودة (Quantity) .
- ١٤- السماح للمستخدم بتكرار هذه العملية لأى جزء آخر .
- ١٥- اذا كانت هناك سجلات قد سبق تجهيزها للمسح يتم اثناء عملية المسح باستخدام الأمر (PACK) .
- ١٦- يتم الرجوع الى برنامج تشغيل ملف المبيعات (SMenu.prg) .

١٧ - ٢ - ٢ كتابة البرنامج

يتم كتابة سطور البرنامج كالآتى :

```
*****SalEdit.prg
*   Edit the Sales File and update Master.
*   Called from Sales Menu , SMenu.prg.

*   Set up memory variables.
No_Dels = 0
Search = 1

* - - - - Set up loop for invoice numbers .
DO WHILE Search # 0
    CLEAR
    @ 2,1 SAY "Edit Sales Transactions"
```

```

@ 2,60 SAY DTOC(T_Date) + " " + TIME()
@ 3,0 SAY Uline
?
?
@ 15,5 SAY "Enter invoice number (0 if none) : " ;
GET Search PICT "9999"
READ

* - - - - If user does not request exit, continue
* - - - - with edit.
IF Search > 0
USE Sales INDEX Sales
* - - - Count records with that invoice number.
COUNT FOR invoice_no = Search TO Howmany
DO CASE
    * - - - If invoice not found , warn user.
    CASE Howmany = 0
        @ 24,1 SAY "No such invoice number"
        ? CHR(7)
    * - - - -If invoice number found , proceed.
    CASE Howmany > 0
        * - - If several records have this invoice
        * - - number , display them and get the
        * - - required record number
        IF Howmany > 1
            @ 5,1 CLEAR
            ? "RecNo Part # Qty Price Clerk"
            ?? " Customer Date"
            ?
            LIST FOR Invoice_no = Search Part_no , ;
            Qty, Price , Clerk, Customer , Date
            ?
            INPUT "Edit which record:" TO RecNo
            GOTO RecNo
        * - - - Otherwise, just go to the record.
    ELSE
        LOCATE FOR Invoice_no = Search
ENDIF

```

```

* Store original field values to variables.
Old_Part = Part_no
Old_Qty = Qty
* - - Display data on edit screen and allow
* - - edit.
@ 5,0 CLEAR
Deleted = " "
@ 7,1 SAY "Invoice Number" GET Invoice_no
@ 7,44 SAY "Date" GET Date PICT "99/99/99"
@ 10,1 SAY "Part Number" GET Part_no;
    PICT "!!!!!"
@ 12,1 SAY "Clerk" GET Clerk
@ 12,40 SAY "Customer" GET Customer
@ 15,1 SAY "Qauntity" GET Qty
@ 15,20 SAY "Selling price" GET Price
@ 24,1 SAY "Delete this record? (Y/N)" ;
    GET Deleted PICT "!"
READ
* - - - After editing , adjust Master file.
DO CASE
    * - - - If transaction to be deleted ,
    * - - - delete it and add its quantity
    * - - - back to the Master file.
    CASE Posted .AND. Deleted = "Y"
        DELETE
        No_Dels = No_Dels + 1
        USE Master INDEX Master
        SEEK Old_Part
        IF FOUND()
            REPLACE Qty WITH Qty + Old_Qty
        ENDIF
        * - If Part number changed , add
        * - quantity to old part number,
        * - and subtract it from new part
        * - - - number.
        CASE Posted .AND. Part_no # Old_Part
            New_Qty = Qty
            New_Part = Part_no

```

```

        USE Master INDEX Master
        SEEK Old_Part
        IF FOUND()
            REPLACE Qty WITH Qty + old_Qty
        ENDIF
        * - -If quantity changed
        * - -adjust the Master file quantity
        CASE Posted .AND. Qty # Old_Qty
            Diff = Qty - Old_Qty
            USE Master INDEX Master
            SEEK Old_Part
            IF FOUND()
                REPLACE Qty WITH Qty - Diff
            ENDIF
            SEEK New_Part
            IF FOUND()
                REPLACE Qty WITH Qty - New_Qty
            ENDIF
        ENDCASE(adjustments after editing)
    ENDCASE(Howmany > 0)
    ENDIF(Search > 0)
    ENDDO(while still editing)

    * ~ ~ ~ -If there are records to be deleted , Pack the
    * ~ ~ ~ -sales database file.
    IF No_Dels > 0
        @ 5,0 CLEAR
        ? "Deleting unwanted records from the sales file.."
        USE Sales INDEX Sales
        PACK
    ENDIF

    * ~ ~ ~ ~ ~ Return to Sales menu.
    CLOSE DATABASES
    RETURN
    
```

والبرنامج يبدأ بإنشاء متغير الذاكرة (No_Dels) لتخزين عدد السجلات المطلوب مسحها ، وكذلك إنشاء متغير الذاكرة (Search) لتخزين رقم الفاتورة التي يريد المستخدم تعديلها . ويتم ذلك من خلال السطور التالية :

```
No_Dels = 0
Search = 1
```

ثم يتم تكوين حلقة تكرارية لسؤال المستخدم عن رقم الفاتورة المطلوب تعديلها . وذلك من خلال السطور التالية :

```
DO WHILE Search # 0
CLEAR
@ 2,1 SAY "Edit Sales Transactions"
@ 2,60 SAY DTOC(T_Date) + " " + TIME()
@ 3,0 SAY Uline
?
?
@ 15,5 SAY "Enter invoice number (0 if none) :";
GET Search PICT "9999"
READ
```

ثم يتم فتح ملف البيعات (Sales.dbf) والفهرس الخاص به للبحث عن رقم الفاتورة الذي أدخله المستخدم (في حالة ادخال هذا الرقم) . ويتم حصر عدد السجلات التي تشترك في هذا الرقم وتخزين هذا العدد في المتغير (Howmany) . ويتم ذلك من خلال السطور التالية :

```
IF Search > 0
USE Sales INDEX Sales
* - - - - - Count records with that invoice number.
COUNT FOR invoice_no = Search TO Howmany
```

إذا لم يكن هناك أى سجلات بنفس رقم الفاتورة يتم تحذير المستخدم وذلك من خلال السطور التالية :

```
DO CASE
* - - - - - If invoice not found , warn user.
CASE Howmany = 0
```

```
@ 24,1 SAY "No such invoice number"
? CHR(7)
```

إذا كان هناك عدة سجلات لها نفس رقم الفاتورة يقوم البرنامج بعرض بيانات هذه السجلات حتي يقوم المستخدم بتحديد رقم السجل المطلوب تعديله . ثم يتم الذهاب الى هذا السجل لتعديله . ويتم ذلك من خلال السطور التالية :

```
CASE Homany > 0
* - - - If several records have this invoice
* - - - number , dispaly them and get the
* - - - required record number
IF Howmany > 1
@ 5,1 CLEAR
? "RecNo Part # Qty Price Clerk"
?? "Customer Date"
?
LIST FOR Invoice_no = Search Part_no , ;
Qty, Price , Clerk, Customer , Date
?
INPUT "Edit which record:" TO RecNo
GOTO RecNo
```

إذا كان هناك سجل واحد برقم الفاتورة الذي تم ادخاله يتم الذهاب الى هذا السجل . وذلك كالآتي :

```
ELSE
LOCATE FOR Invoice_no = Search
ENDIF
```

وفي الجزء التالي يتم تخزين رقم الجزء الأصلي والكمية الأصلية الموجودة في السجل الذي تم اختياره في متغيرات ذاكرة قبل أن يقوم المستخدم بتعديلها . وذلك حتي يمكن اختبار هذه الحقول بعد ذلك . ويتم ذلك من خلال السطور التالية :

```
Old_Part = Part_no
Old_Qty = Qty
```


وفي الجزء التالي يتم انشاء متغير الذاكرة (Deleted) لتخزين رغبة المستخدم في مسح السجل أو عدم مسحه . ثم يتم تصميم شاشة لتصحيح بيانات السجل . ويتم ذلك من خلال السطور التالية :

```
@ 5,0 CLEAR
Deleted = " "
@ 7,1 SAY "Invoice Number" GET Invoice_no
@ 7,44 SAY "Date" GET Date PICT "99/99/99"
@ 10,1 SAY "Part Number" GET Part_no ;
    PICT "!!!!!"
@ 12,1 SAY "Clerk" GET Clerk
@ 12,40 SAY "Customer" GET Customer
@ 15,1 SAY "Qauntity" GET Qty
@ 15,20 SAY "Selling price" GET Price
@ 24,1 SAY "Delete this record? (Y/N)" GET ;
    Deleted PICT "!"
READ
```

وفي الجزء التالي من البرنامج يتم اختبار هذا السجل بعد تعديله بواسطة المستخدم . فإذا كان السجل قد تم ترحيله (Posted) وطلب المستخدم مسحه (Deleted = "Y") يتم تجهيزه للمسح بواسطة الأمر (DELETE) ثم يتم زيادة عدد السجلات المطلوب مسحها (No_Dels) بواحد .

ثم يقوم البرنامج بعد ذلك بفتح الملف الرئيسي (Master) وملف الفهرس الخاص به للبحث عن هذا السجل المطلوب مسحه وإضافة الكمية التي عدلها المستخدم مرة ثانية . ويتم ذلك من خلال السطور التالية :

```
DO CASE
    * - - - - - If transaction to be deleted ,
    * - - - - - delete it and add its quantity
    * - - - - - back to the Master file.
CASE Posted .AND. Deleted = "Y"
DELETE
No_Dels = No_Dels + 1
USE Master INDEX Master
```

```

SEEK Old_Part
IF FOUND()
    REPLACE Qty WITH Qty + Old_Qty
ENDIF

```

وفي الجزء التالي يتم اختبار حالة ثانية في السجل ، وهي عندما يقوم المستخدم بتعديل رقم الجزء . وذلك عندما يكتشف مثلا أنه أدخل رقم أحد الأجزاء خطأ عند كتابة فاتورة البيع . وفي هذه الحالة تكون كمية الصنف (Quantity) الخاصة بهذا الجزء قد نقصت في الملف الرئيسي (Master) في حين تظل كمية الصنف الذي كان يجب ادخال رقمه كما هي . ولذلك يقوم البرنامج بإضافة الكمية التي سبق خصمها من الصنف في الملف الرئيسي الى الجزء القديم (Old_Part) وخصمها من كمية الصنف في الجزء الجديد . ويتم ذلك من خلال السطور التالية :

```

CASE Posted .AND. Part_no # Old_Part
    New_Qty = Qty
    New_Part = Part_no
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty + Old_Qty
    ENDIF

```

والجزء التالي يعالج الحالة الثالثة وهي عندما يعدل المستخدم الكمية (Quantity) فقط دون تعديل رقم الجزء . في هذه الحالة يقوم البرنامج بتعديل كمية هذا الجزء في الملف الرئيسي (Master) ، وذلك بخصم الكمية القديمة من الكمية الجديدة التي عدلها المستخدم وتخزين هذا الفرق في المتغير (DIFF) . ثم يتم طرح هذا الفرق (DIFF) من الكمية الحالية الموجودة في الملف الرئيسي ويتم ذلك من خلال السطور التالية :

```

CASE Posted .AND. Qty # Old_Qty
    Diff = Qty - Old_Qty
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty - Diff
    ENDIF
ENDCASE(adjustments after editing)

```

```
ENDCASE(Howmany > 0)
ENDIF(Search > 0)
ENDDO(while still editing)
```

وفي الجزء التالي يتم اختبار المتغير (No_Dels) لمعرفة ما إذا كان هناك سجلات مطلوب مسحها أم لا . فإذا كانت هناك سجلات مطلوب مسحها يتم استخدام الأمر (PACK) في مسحها نهائيا من ملف البيعات (Sales.dbf) . ويتم ذلك من خلال السطور التالية :

```
IF NO_Dels > 0
  @ 5,0 CLEAR
  ? "Deleting unwanted records from the sales file.."
  USE Sales INDEX Sales
  PACK
```

```
ENDIF
```

وفي نهاية البرنامج يتم اغلاق الملفات والعودة الى برنامج المبيعات . (SMenu.prg)

١٧ - ٢ برنامج تصحيح ملف الاضافة (NewEd.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (3) في قائمة برنامج الاضافة . انظر الشكل (١٧ - ٢) .

<u>New stock system menu</u>		<u>02/20/90</u>	<u>08:30:45</u>
<ol style="list-style-type: none"> 1. Record new items 2. Print new stock data 3. Edit new stock data 4. Return to main menu 			
Enter choice (1 - 4)			

شكل (١٧ - ٢)

وهذا البرنامج مثل البرنامج السابق يسمح للمستخدم بتعديل سجلات اضافة الاصناف التي سبق ادخالها ، بحيث يتم تعديل الملف الرئيسي (Master) بناء على هذه التعديلات اذا كانت السجلات قد سبق ترحيلها (Posted) . وهو لا يختلف في تركيبه عن البرنامج السابق ولكنه يختلف عنه في الحاجة الى تعديل حقل سعر الصنف (Cost) وحقل (On_order) بالاضافة الى حقل الكمية (Qty) . كما ان البرنامج يستخدم رقم الجزء في البحث عن الصنف المطلوب في ملف الاضافة (NewStock.dbf) وليس رقم الفاتورة مثل ملف المبيعات . ويتكون البرنامج من السطور التالية :

```
*****NewEdit.prg
*   Edit the NewStock File and update Master.
*   Called from NMenu.prg.
*   Set up memory variables.
No_Dels = 0
Search = "1"

* - - - - Get part number for data to edit.
DO WHILE Search # "0"
    Use NewStock INDEX NewStock
    CLEAR
    @ 2,1 SAY "Edit NewStock Transactions"
    @ 2,60 SAY DTC(T_Date) + "    " + TIME()
    @ 3,0 SAY Uline
    ?
    ACCEPT "Enter part number to edit (0 if none)" ;
    TO Search
    Search = UPPER(Search)
    * - - - If user did not request exit, continue.
    IF Search # "0"
        * - - Count the records with that part number
        SEEK Search
        COUNT WHILE Part_no = Search TO Howmany
        DO CASE
            * - - If part not found , warn user.
            CASE Howmany = 0
                ? " No such part number"
                ? CHR(7)
```

```

* - - - If part number found , proceed.
CASE Howmany > 0
  * - - If several records have that part
  * - - number , display them and get the
  * - - required record
  IF Howmany > 1
    CLEAR
    SEEK Search
    LIST WHILE Part_no = Search Part_no , ;
      Qty, Cost, Date, Vendor
    ?
    INPUT "Edit which record(enter ;
      record number)?" TO RecNo
    GOTO RecNo
  ELSE
    SEEK Search
    RecNo = RECNO()
  ENDIF(Howmany > 1)

  * Store original field values to variables
  Old_Part = Part_no
  Old_Qty = Qty
  Old_Cost = Cost
  *- - Display edit screen and allow edit.
  CLEAR
  Deleted = " "
  @ 1,1 SAY "Edit NewStock Transaction"
  @ 3,1 SAY "Part Number" GET Part_no ;
  PICT "!!!!!"
  @ 5,1 SAY "Quantity" GET Qty
  @ 5,20 SAY "Purchase Price" GET Cost
  @ 7,1 SAY "Date" GET Date PICT "99/99/99"
  @ 7,15 SAY "Vendor" GET Vendor
  @ 9,2 SAY "Delete this record? (Y/N)" ;
  GET Deleted PICT "!"
  READ

  * - After editing , adjust Master file.

```

```

* - - First handle change in cost.
IF Cost # Old_Cost
    New_Part = Part_no
    New_Cost = Cost
    USE Mater INDEX Master
    SEEK New_Part
    IF FOUND()
        REPLACE Cost WITH New_Cost
        USE NewStock INDEX NewStock
        GOTO RecNo
    .ENDIF
ENDIF (Cost # Old_Cost)
DO CASE
    * - - -if new stock transaction to
    * - - -be deleted, subtract its
    * - - -quantity from the Master
    * - - -file and add it to the
    * - - -On_order field.
    CASE Posted .AND. Deleted = "Y"
        DELETE
        No_Dels = No_Dels + 1
        USE Master INDEX Master
        SEEK Old_Part
        IF FOUND()
            REPLACE Qty WITH Qty + ;
            Old_Qty
            REPLACE On_order WITH ;
            On_order + Old_Qty
        ENDIF
        * - - If Part number changed,
        * - - subtract quantity from the old
        * - - part number, and add to the
        * - - new . Do the opposit for the
        * - - On_order field.
        CASE Posted .AND. Part_no # Old_Part
            New_Qty = Qty
            New_Part = Part_no
            USE Master INDEX Master

```

```
SEEK Old_Part
IF FOUND()
    REPLACE Qty WITH Qty - Old_Qty
    REPLACE On_order WITH On_order;
    + Old_Qty
ENDIF
SEEK New_Part
IF FOUND()
    REPLACE Qty WITH Qty + New_Qty
    REPLACE On_order WITH On_order;
    - New_Qty
ENDIF
* - - If user just changed the
* - - quantity , adjust the
* - - Master file quantity.
CASE Posted .AND. Qty # Old_Qty
    Diff = Qty - Old_Qty
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty + Diff
        REPLACE On_order WITH ;
        On_order - Diff
    ENDIF
ENDCASE
ENDCASE
ENDIF(Search # 0).
ENDDO(while Search # 0)

* - - - If records have been deleted , Pack the New
* - - - - Stock data file.
IF NO_Dels > 0
    CLEAR
    ? "Deleting unwanted records from NewStock file"
    USE NewStock INDEX NewStock
    PACK
ENDIF
```

```
* - - - - Done , Return to NewStock Menu.
CLOSE DATABASES
RETURN
```

والبرنامج يبدأ بإنشاء متغيرات الذاكرة مثل البرنامج السابق تماما . ولكن يلاحظ أن المتغير (Search) يتم إنشاؤه كمتغير حرفي ، وذلك لأن البحث سوف يتم من رقم الجزء وليس رقم الفاتورة مثل البرنامج السابق . ورقم الجزء يمكن أن يحتوي على أرقام وحروف . ولذلك تستخدم الدالة (UPPER) في تحويل رقم الجزء الذي يدخله المستخدم إلى حروف كبيرة حتى يماثل الفهرس . والسطور التالية توضح هذا الجزء

```
No_Dels = 0
Search = "1"
```

```
* - - - - Get part number for data to edit.
DO WHILE Search # "0"
    Use NewStock INDEX NewStock
    CLEAR
    @ 2,1 SAY "Edit NewStock Transactions"
    @ 2,60 SAY DTOC(T_Date) + " " + TIME()
    @ 3,0 SAY Uline
    ?
    ACCEPT "Enter part number to edit(0 if none)" TO;
    Search
    Search = UPPER(Search)
```

ثم يتم البحث عن رقم الجزء الذي أدخله المستخدم وحصر عدد السجلات التي لها هذا الرقم وتخزين هذا العدد في المتغير (Howmany) . ويتم ذلك من خلال السطور التالية :

```
IF Search # "0"
* - - - - Count the records with that part number
SEEK Search
COUNT WHILE Part_no = Search TO Howmany
```

وإذا لم يتم العثور على هذا الرقم يتم تحذير المستخدم وذلك كالآتي :


```
DO CASE
  * - - - - - If part not found , warn user.
  CASE Howmany = 0
  ? " No such part number"
  ? CHR(7)
```

وإذا كان هناك عدة سجلات لها نفس رقم الجزء المطلوب ، يتم عرض بيانات هذه السجلات وسؤال المستخدم عن رقم السجل المطلوب . ويتم ذلك من خلال السطور التالية :

```
IF Howmany > 1
  CLEAR
  SEEK Search
  LIST WHILE Part_no = Search Part_no , ;
  Qty, Cost, Date, Vendor
  ?
  INPUT "Edit which record(enter record ;
  number) ?" TO RecNo
  GOTO RecNo
```

وإذا كان هناك سجل واحد يحتوى على رقم الجزء المطلوب يتم الذهاب الى هذا السجل وتخزين رقم السجل في متغير الذاكرة (RecNo) ويتم ذلك من خلال السطور التالية :

```
ELSE
  SEEK Search
  RecNo = RECNO()
ENDIF(Howmany > 1)
```

وفي الجزء التالى يتم تخزين محتويات بعض حقول السجل المطلوب تعديله في متغيرات ذاكرة حتى يمكن استخدامها بعد ذلك في تعديل الملف الرئيسى . ويتم ذلك من خلال السطور التالية :

```
old_part = part_no
old_Qty = Qty
old_Cost = Cost
```

وفي الجزء التالي يقوم البرنامج بعرض بيانات سجل الاضافة للمستخدم حتى يمكنه تعديل الحقول المطلوبة أو مسح السجل من ملف الاضافة (NewStock.dbf) حسب الحاجة . ويتم ذلك من خلال السطور التالية :

```
CLEAR
Deleted = " "
@ 1,1 SAY "Edit NewStock Transaction"
@ 3,1 SAY "Part Number" GET Part_no ;
  PICT "!!!!!"
@ 5,1 SAY "Quantity" GET Qty
@ 5,20 SAY "Purchase Price" GET Cost
@ 7,1 SAY "Date" GET Date PICT "99/99/99"
@ 7,15 SAY "Vendor" GET Vendor
@ 9,2 SAY "Delete this record? (Y/N)" GET ;
  Deleted PICT "!"
READ
```

وفي الجزء التالي يقوم البرنامج بضبط البيانات في الملف الرئيسي (Master.dbf) بناء على البيانات التي تم تعديلها في ملف الاضافة (NewStock.dbf) . فاذا كان المستخدم عدل سعر الجزء (Cost) يتم تعديله في الملف الرئيسي . وذلك من خلال السطور التالية :

```
IF Cost # Old_Cost
  New_Part = Part_no
  New_Cost = Cost
  USE Mater INDEX Master
  SEEK New_Part
  IF FOUND()
    REPLACE Cost WITH New_Cost
    USE NewStock INDEX NewStock
    GOTO RecNo
  ENDIF
ENDIF(Cost # Old_Cost)
```

وفي الجزء التالي يتم اختبار حالة أخرى ، عندما يطلب المستخدم مسح هذا السجل (Deleted = "Y") ، فان البرنامج يقوم بتجهيزه للمسح باستخدام الأمر (DELETE) ، ثم يقوم بفتح الملف الرئيسي والفهرس الخاص به . ويتم البحث عن

هذا السجل فاذا كان موجودا في الملف الرئيسي يتم خصم الكمية التي سبق اضافتها من الملف الرئيسي ، كما يتم اضافة هذه الكمية الى الكمية تحت الطلب (On_order) . وذلك من خلال السطور التالية :

```
DO CASE
CASE Posted .AND. Deleted = "Y"
    DELETE
    No_Dels = No_Dels + 1
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty - Old_Qty
        REPLACE On_order WITH ;
        On_order + Old_Qty
    ENDIF
```

والجزء التالي يعالج حالة أخرى عندما يقوم المستخدم بتعديل رقم الجزء الذي سبق الاضافة عليه . وفي هذه الحالة يقوم البرنامج بخصم الكمية المضافة من الجزء السابق و اضافتها الى الجزء الجديد . كما يتم اضافة هذه الكمية الى الكمية تحت الطلب . ويتم ذلك من خلال السطور التالية :

```
CASE Posted .AND. Part_no # Old_Part
    New_Qty = Qty
    New_Part = Part_no
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty - Old_Qty
        REPLACE On_order WITH On_order;
        + Old_Qty
    ENDIF
    SEEK New_Part
    IF FOUND()
        REPLACE Qty WITH Qty - New_Qty
        REPLACE On_order WITH On_order;
        - New_Qty
    ENDIF
```

والجزء التالي يعالج الحالة الأخيرة عندما يقوم المستخدم بتعديل الكمية التي سبق اضافتها . في هذه الحالة يتم ضبط الكمية في الملف الرئيسي بناء على الفرق بين الكمية الأولى والكمية الجديدة بعد التعديل . والسطور التالية توضح هذه العملية :

```
CASE Posted .AND. Qty # Old_Qty
  Diff = Qty - Old_Qty
  USE Master INDEX Master
  SEEK Old_Part
  IF FOUND()
    REPLACE Qty WITH Qty + Diff
    REPLACE On_order WITH ;
      On_order - Diff
  ENDIF
ENDCASE
ENDCASE
ENDIF(Search # 0)
ENDDO(while Search # 0)
```

وفي الجزء التالي يتم مسح السجلات التي سبق تجهيزها للمسح . ويتم ذلك باستخدام الأمر (PACK) . ثم يتم اغلاق الملفات والعودة الى برنامج الاضافة (NMenu.prg) . والسطور التالية توضح هذه العملية :

```
IF NO_Dels > 0
  CLEAR
  ?"Deleting unwanted records from the NewStock file"
  USE NewStock INDEX NewStock
  PACK
ENDIF
CLOSE DATABASES
RETURN
```

4

الجزء الرابع

نظام حسابات العملاء

- * تصميم النظام
- * ملفات الخطوات
- * برنامج القائمة الرئيسية والادخال والتعديل
- * تقارير برنامج حسابات العملاء
- * التحديث الشهري للنظام

الفصل الثامن عشر

تصميم النظام

يتم في هذا الجزء تصميم نظام حسابات العملاء (Accounts Receivable) الذي يشتمل على طباعة الفواتير آلياً . ويتم في هذا النظام تخزين اسم العميل (Customer) وعنوانه في ملف قاعدة بيانات منفصل بالإضافة الى الموازنة الحالية (Current balance) والموازنات السابقة (Aged balances) . كما يتم تخزين بيانات السداد (Payments) في ملف قاعدة بيانات منفصل . كما يتم تخزين بيانات الصرف (Charges) في ملف قاعدة بيانات آخر ويقوم البرنامج بالربط بين هذه الملفات الثلاثة .

ودراسة هذا النظام تتيح لمخطط البرامج معرفة وسائل جديدة للتعامل مع الملفات المرتبطة (Relational Databases) . كما يتم من خلالها دراسة استخدام ملفات الخطوات (Procedure Files) في تسهيل كتابة البرامج وزيادة كفاءتها .

١٨ - ١ تعريف المشكلة

الهدف من نظام حسابات العملاء هو انشاء قاعدة بيانات يتم من خلالها متابعة بيانات العملاء ، وتسجيل الموقف الشهري (Monthly) لحركة السداد (Payments Transactions) وحركة الصرف (Charges Transactions) لكل حساب (Account) . ويقوم النظام باصدار فواتير (Invoices) لكل حساب توضح الموازنة الابتدائية (Starting Balance) للشهر ، وكل حركة سداد أو صرف خلال هذا الشهر ، ثم الموازنة الحالية (Current Balance) . كما يوفر النظام تقارير مختصرة (Summary Reports) للنشاط الشهري لكل عميل . كما يوفر أيضا تقارير للموازنة خلال ٣٠ يوما ، ٦٠ يوما ، ٩٠ يوما وأكثر من ٩٠ يوما . كما يتيح النظام أيضا تخزين نسخة من فاتورة العميل حتى يستطيع العميل مراجعة أي بيانات سابقة .

ويقوم البرنامج كالمادة باستخدام القوائم الواضحة في تسهيل تعامل المستخدم مع النظام .

١٨ - ٢ تحديد هيكل قاعدة البيانات

كما سبق الايضاح فان قاعدة البيانات تتكون من ثلاثة ملفات . اولها ملف بيانات العميل (Customer) الذي يحتوى على كل البيانات الأساسية لهذا العميل . والثاني ملف حركة الصرف (Charges) الذي يتضمن بيانات فواتير الصرف والأجزاء المصروفة وأسعارها . والثالث ملف حركة السداد (Payments) الذي يتضمن

بيانات الشيكات وموقف تسديد الفواتير .

١٨ - ٢ - ١ ملف بيانات العميل (Customer.dbf)

يتكون ملف بيانات العميل من الحقول الموضحة بالشكل (١٨ - ١)

Field	Field Name	TYPE	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	CUST_NAME	Character	25	
3	ADDRESS	Character	25	
4	PHONE	Character	13	
5	LAST_UPDAT	Date	8	
6	START_BAL	Numeric	8	2
7	CHG_CURR	Numeric	8	2
8	PAY_CURR	Numeric	8	2
9	BAL_30	Numeric	8	2
10	BAL_60	Numeric	8	2
11	BAL_90	Numeric	8	2
12	BAL_90, PLUS	Numeric	8	2
13	TERMS	Character	20	

شكل (١٨ - ١)

والحقل رقم (١) يمثل رقم الحساب وهو حقل عددي . والحقل رقم (٢) يمثل اسم العميل وهو حقل حرفي يتكون من ٢٥ حرفا . والحقل رقم (٣) يمثل العنوان وهو حقل حرفي يتكون من ٢٥ حرفا . والحقل رقم (٤) هو حقل رقم التليفون وهو حقل حرفي يتكون من (١٢) حرفا . والحقل رقم (٥) هو حقل تاريخ آخر تعديل . والحقل رقم (٦) هو حقل الموازنة الابتدائية وهو حقل عددي مكون من ثمانية أرقام ورقمين عشريين . والحقل رقم (٧) هو حقل حركة الصرف الحالية (Current Charges) . والحقل رقم (٨) هو حقل حركة السداد

الحالية (Current Payment) . والحقل رقم (١) هو حقل الموازنة خلال شهر . والحقل رقم (١٠) هو حقل الموازنة خلال شهرين . والحقل رقم (١١) هو حقل الموازنة خلال ثلاثة شهور . والحقل رقم (١٢) هو حقل الموازنة خلال أربعة أشهر . والحقل رقم (١٣) هو حقل الدين (Credit) .

ويستخدم حقل رقم الحساب (Account_No) في ربط الملفات الثلاثة . ولذلك فمن الطبيعي أن تتم فهرسة الملفات الثلاثة بناء على هذا الحقل . ولفهرسة ملف العميل (Cnstomer. dbf) يتم كتابة السطر التالي من مشيرة النقطة (Dot Prompt) .

INDEX ON Account_No TO Cust_No

ولكن قد يحتاج المستخدم في بعض الأحيان إلى البحث عن عميل معين باسمه ، لذلك يمكن إنشاء فهرس آخر بناء على حقل اسم العميل (Cust_name) . ويمكن استخدام الدالة (UPPER) في تحويل اسم العميل في الفهرس إلى حروف كبيرة (Uppercase) لتسهيل البحث عن اسم العميل سواء أدخل المستخدم الاسم بحروف صغيرة أو حروف كبيرة ، ولإنشاء هذا الفهرس يتم كتابة السطر التالي من مشيرة النقطة (Dot Prompt) .

INDEX ON UPPER(Cust_name) TO Cust_Name

١٨ - ٢ - ٢ ملف حركة الصرف (Charges.dbf)

يتكون ملف حركة الصرف (Charges.dbf) من الحقول الموضحة في الشكل (١٨ - ٢) . ويلاحظ أن حقل رقم الحساب (Account no) موجود في الملف بنفس الاسم ونفس النوع (Type) ونفس العرض (Width) ، وذلك لأن الحقل الذي سوف يستخدم كفتاح (Key Field) لربط الملفات الثلاثة .

ويتم إنشاء فهرس لهذا الملف بناء على حقل رقم الحساب ، وذلك بكتابة السطر التالي من مشيرة النقطة (Dot Prompt) .

INDEX ON Account_No TO ChrgNo

Field	Field Name	TYPE	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	INVOICE_NO	Numeric	6	0
3	PART_NO	Character	5	
4	QTY	Numeric	7	2
5	UNIT_PRICE	Numeric	9	2
6	AMOUNT	Numeric	9	2
7	DATE	Date	8	
8	DESCRIPT	Character	20	
9	BILLED	Logical	1	

شكل (١٨ - ٢)

والحقل رقم (١) يمثل رقم الحساب ، والحقل رقم (٢) يمثل رقم الفاتورة ، والحقل رقم (٣) يمثل حقل رقم الجزء ، والحقل رقم (٤) يمثل حقل الكمية ، والحقل رقم (٥) يمثل حقل سعر الوحدة . والحقل رقم (٦) يمثل حقل الكمية الكلية المصروفة ، والحقل رقم (٧) يمثل حقل تاريخ الشراء ، والحقل رقم (٨) يمثل حقل وصف عملية الشراء ، والحقل رقم (٩) هو حقل منطقي يوضح إذا كان الحساب قد تم تسديده أم لا .

١٨ - ٢ - ٣ ملف السداد (Payments.dbf)

أما ملف السداد (Payments.dbf) فيتم تكوينه من الحقول الموضحة في الشكل (١٨ - ٣) .

والحقل رقم (١) يمثل رقم الحساب ، والحقل رقم (٢) يمثل رقم الشيك ، والحقل رقم (٣) يمثل حقل الكمية الكلية المدفوعة ، والحقل رقم (٤) يمثل حقل تاريخ الشراء ، والحقل رقم (٥) يمثل حقل وصف العملية ، والحقل رقم (٦) هو حقل منطقي يوضح إذا كان الدفع قد تم تسجيله أم لا .

Field	Field Name	TYPE	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	CHECK_NO	Character	5	
3	AMOUNT	Numeric	9	2
4	DATE	Date	8	
5	DESCRIPT	Character	30	
6	POSTED	Logical	1	

شكل (١٨ - ٢)

ويتم انشاء ملف الفهرس لهذا الملف بكتابة السطر التالى من مشيرة النقطة (Dot Prompt) كالآتى :

INDEX ON Cust_no TO PayNo

١٨ - ٣ حفظ البيانات التاريخية

يقوم نظام حسابات العملاء كما سبق الايضاح على المتابعة الشهرية ، أى تسجيل حركة المصرف والتوريد خلال الشهر . وفى نهاية الشهر يتم تجميع الفواتير وارسالها الى العملاء .

ولكن ماذا لو أراد المستخدم الرجوع الى بيانات سابقة عن حسابات تم تسديدها؟

هناك ثلاثة احتمالات للتعامل مع البيانات القديمة :

- ١- مسح كل حركة للمصرف أو السداد بمجرد تسديدها . وتمتاز هذه الطريقة بتوفير مساحة تخزينية على القرص ، ولكن يميها عدم القدرة على الرجوع الى أى بيانات سابقة عن أى حركة تم تسديدها .

- ٢ - ترك كل حركة صرف أو سداد مخزنة في الملف حتى بعد تسديدها، مع تمييز الحركة التي تم تسديدها . وفائدة هذه الطريقة أنها تحتفظ بالبيانات السابقة عن أي حركة مهما كان تاريخها . ولكن يميزها أنها سوف تؤدي إلى تضخم ملفي الصرف (Charges) والسداد (Payment) إلى حجم لا نهائي .
- ٢ - نقل كل حركة صرف أو سداد بمجرد تسديدها من ملفي الصرف والسداد إلى ملف تاريخي (Historical file) . وفائدة هذه الطريقة أنها تؤدي إلى الاحتفاظ بملفي الصرف والتوريد صغيرين نسبياً مع الاحتفاظ بجميع البيانات السابقة عن أي حركة سبق تسديدها .

ومن خلال نظام حسابات العملاء الجاري شرحه سيتم استخدام الاحتمال الثالث ، وهو نقل جميع السجلات التي تم تسديدها من ملفي الصرف والسداد إلى ملف تاريخي (Historical File) . ويمكن تخزين هذا الملف التاريخي في قرص أو أقراص منفصلة . ولإنشاء ملف تاريخي لملف الصرف (Charges.dbf) يتم نسخ هيكل الملف (Structure) إلى ملف جديد يتم تسميته (BillHist.dbf) مثلاً . وذلك بكتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) .

```
USE Charges
COPY STRUCTURE TO BillHist
```

ولإنشاء ملف تاريخي مماثل بالنسبة لملف السداد (Payments) يتم كتابة السطرين التاليين من مشيرة النقطة .

```
USE Payments
COPY STRUCTURE TO PayHist
```

وفي الأجزاء التالية سيتم إيضاح كيفية نقل السجلات التي تصبح غير مطلوبة إلى الملفات التاريخية من خلال برنامج حسابات العملاء .

١٨ - ٤ تركيب البرنامج

يتكون برنامج حسابات العملاء من عدة برامج منفصلة يقوم أحدها بإضافة عملاء جدد ، والثاني بطباعة التقارير والفواتير ، والثالث بتعديل بيانات أحد العملاء ، والرابع بالترحيل الشهري و ... الخ .

والشكل الهرمي (Hierarchical) للبرنامج يتضح من الشكل (١٨ - ٤) .
ويلاحظ من الشكل أن البرنامج أكثر تمقيداً من برنامج المخازن حيث أنه يشمل عدة
ملفات . ولذلك سيتم في الأجزاء التالية استخدام وسيلة جديدة لزيادة سرعة تنفيذ
البرنامج وتقليل عدد الملفات المستخدمة . وهذه الوسيلة هي استخدام ملف
الخطوات (Procedure File) وسيتم شرحها في الباب التالي .



شكل (١٨ - ٤)

الفصل التاسع عشر

ملفات الخطوات

في هذا الفصل سيتم شرح احدي الوسائل المتقدمة لكتابة البرامج بواسطة (DBase III+) أو برامج عائلة (DBase) الأخرى ، وهي استخدام ملفات الخطوات (Procedure Files) والمعاملات (Parameters) . وسوف يتم استخدام هذه الوسيلة في برنامج حسابات العملاء (A/R) .

ملف الخطوات هو ملف يحتوي على برامج صغيرة (Routines) ، ويمكن استخدام كل برنامج في عدة أماكن من البرنامج الرئيسي باستدعائها عند الحاجة إليها أي أنه يعتبر مكتبة برامج (Library) . ويمكن إنشاء عدة ملفات خطوات . ولكن لا يمكن فتح أكثر من ملف في نفس الوقت . وهي تؤدي إلى تسهيل كتابة البرامج وسرعة تنفيذها كما أنها تتغلب على مشكلة زيادة عدد الملفات المفتوحة التي تظهر في البرامج الكبيرة . وذلك لأن ملف الخطوات يعتبر ملفاً واحداً رغم أنه يحتوي على عدة برامج .

ارجع الى الأمر (SET PROCEDURE TO) في الكتاب السابق .

١٩ - ١ استخدام ملف الخطوات في برنامج حسابات العملاء

يستخدم مع برنامج حسابات العملاء ملف خطوات واحد يحتوي على ثلاثة برامج . وهذا الملف عند فتحه باستخدام الأمر (SET PROCEDURE TO) يتم تحميله في الذاكرة المؤقتة (RAM) . وهذا يؤدي إلى سرعة تشغيل البرامج داخله حيث لا تصبح هناك حاجة دائماً إلى نقل هذه البرامج من القرص إلى الذاكرة .

وكل برنامج (Procedure) موجود داخل الملف يمكن تحويله إلى برنامج عام (General) يمكن استخدامه في أي برامج أخرى . ويتم ذلك باستخدام الأمر (PARAMETERS) . وهذا الأمر يؤدي إلى ادخال معاملات معينة إلى البرنامج ليناسب تطبيقاً محدداً . ويمكن استخدام معاملات أخرى لتنفيذ البرنامج في تطبيقات أخرى متعددة . وهذا سيتم أيضاً عند دراسة البرامج الثلاثة (Procedures) المستخدمة مع برنامج حسابات العملاء .

١٩ - ٢ برنامج العنوان (Title)

يستخدم هذا البرنامج لكتابة أي عنوان وكتابة تاريخ اليوم الحالي مع رسم خط عرضي بعرض الشاشة . ويمكن استخدام هذا البرنامج مع أي شاشة يتم تصميمها للمستخدم .

ويبدأ هذا البرنامج بالأمر (PROCEDURE) يليه اسم البرنامج (Title) ، وذلك

في السطر الأول . وفي السطر الثاني يتم كتابة الأمر (PARAMETERS) يليه اسم المعامل أو المعاملات التي سوف يتم استخدامها عند استدعاء البرنامج . والسطر التالي يقوم بمسح الشاشة ، والسطر التالي يقوم بكتابة العنوان (Title) ، والسطر التالي يقوم بكتابة تاريخ اليوم الحالي والوقت الحالي ، والسطر التالي يقوم برسم خط عرضي بعرض الشاشة (Uline) . مع ملاحظة أن المتغير (Uline) يكون قد سبق تعريفه في البرنامج الرئيسي كما تم في برنامج المخازن . وسطور البرنامج التي سبق شرحها تكون كالآتي :

PROCEDURE Title

PARAMETERS Title

```
* - - - - Display screen title
CLEAR
@ 2,1 SAY Title
@ 2,60 SAY DTOC( DATE() + " " + TIME() )
@ 3,0 SAY Uline
?
?
```

RETURN

والشيء المثير في هذه العملية أنه في أي نقطة من البرنامج الرئيسي عندما يراد عرض شاشة للمستخدم متضمنة العنوان وتاريخ اليوم الحالي والوقت والخط العرضي ، فإنه يكفي كتابة الأمر (DO) يليه اسم هذا البرنامج ثم العنوان المراد كتابته . وذلك كالآتي مثلاً :

Do Title WITH "Accounts Receivable Main Menu"

وكلمة (WITH) تنبه برنامج (DBase III+) أن البيانات التالية لها هي المعاملات التي سبق تعريفها في البرنامج الفرعي (Procedure) . ولذلك فعند تنفيذ هذا البرنامج الفرعي ، وعند وصول البرنامج إلى الأمر (PARAMETERS) فإنه يستبدل العنوان (Title) بالعنوان الذي تم ادخاله وهو ("Account Receivable Main Menu") . لذلك فعند تنفيذ البرنامج يظهر الآتي على الشاشة .

Accounts Receivable Main Menu 02/20/90 12:50:30

١٩ - ٣ برنامج رسائل الأخطاء

من البرامج التي تستخدم أيضا في ملفات الخطوات (Procedure Files) برنامج رسائل الأخطاء . وهذا البرنامج يستخدم في عرض رسائل الأخطاء على الشاشة لتوضيح للمستخدم الخطأ الذي وقع فيه والعمل الذي يجب تنفيذه للتغلب على هذا الخطأ . وهذا البرنامج يتكون من السطور التالية :

```
PROCEDURE Error
PARAMETERS Message
* - - - - - Display error message.
@ 20,0 CLEAR
@ 20,3 SAY Message
? CHR(7)
WAIT "Press any key to try again"
RETURN
```

وهذا البرنامج عند استدعائه بواسطة الأمر (DO) يقوم بمسح الشاشة ابتداء من السطر (20) ولأسفل . وذلك لكي تظل البيانات موجودة على الشاشة . ثم يقوم بعرض رسالة الخطأ (Message) التي يتم إدخالها مع الأمر (DO) . كما يشغل الجرس (Bell) ويُنْتَظَر حتى يضغط المستخدم على أى مفتاح أو ينفذ أى شيء يكون موضحا في الرسالة (Message) . فمثلا عند كتابة الأمر التالي :

```
DO Error WITH "No such customer number!"
```

فإن البرنامج يعرض الآتي أسفل الشاشة :

```
No such customer number !
Press any key to try again
```

كما يشغل الجرس (Bell) ويُنْتَظَر من المستخدم الضغط على أى مفتاح لاستكمال تنفيذ البرنامج الرئيسى .

١٩ - ٤ برنامج التحقق من رقم العميل

هذا هو البرنامج الثالث المستخدم بواسطة برنامج حسابات العملاء ، وهو يتيح للمستخدم استعمال اسم العميل أو رقمه في البحث عن بيانات عميل محدد . ويتم

ذلك من خلال أمر واحد كالآتي :

```
DO GetCust WITH ;
M_Cust_No,M_Cust_Name,M_Address,Exiting
```

وهذا الأمر يؤدي الى تنفيذ الآتي :

- ١- عرض شاشة للمستخدم لعرض اسم العميل أو رقمه .
- ٢- إذا أراد المستخدم الخروج يتم إخبار البرنامج القائم بالاستدعاء بذلك .
- ٣- البحث عن أرقام الحساب الخاصة بالاسم الذي تم ادخاله وعرضها على الشاشة .
- ٤- في حالة ادخال رقم حساب غير موجود يتم عرض رسالة خطأ والسماح للمستخدم بالمحاولة مرة ثانية .
- ٥- العودة للبرنامج القائم بالاستدعاء مع اعطائه رقم العميل واسمه وعنوان وعرض هذه البيانات على الشاشة لكي يتحقق المستخدم أن هذا هو العميل المطلوب .

وهذا البرنامج سوف يستخدم كثيرا في برنامج حسابات العملاء . ولذلك فسوف يتم شرحه بالتفصيل في نهاية هذا الفصل .

١٩ - ٥ إنشاء ملف الخطوات

يتم انشاء ملف الخطوات بنفس طريقة انشاء أى ملف أوامر (Command file) . وذلك باستخدام الأمر (MODIFY COMMAND) ثم كتابة اسم الملف ونسبته في هذه الحالة (Proclib1.dbf) وهو اختصار (Procedure Library 1) . وملف الخطوات يمكن أن يحتوى على عدد من البرامج بعد أقصى ٢٢ برنامجا . ولكن في نظام حسابات العملاء فإن الملف (Proclib1) يحتوى على ثلاثة برامج فقط كما سبق الايضاح .

وكل برنامج فرعي (Procedure) يبدأ بالأمر (PROCEDURE) يليه اسم البرنامج وينتهي بالأمر (RETURN) . وفي حالة استخدام المعاملات (Parameters) يتم استخدام الأمر (PARAMETERS) في السطر التالى مباشرة لسطر الأمر (PROCEDURE) .

والسطور التالية توضح أوامر الملف (ProcLib1) .

***** Proclib1.prg

* General Procedure for the AR System

* - - - - - Display screen title.

PROCEDURE Title

PARAMETERS Title

CLEAR

@ 1,0 SAY Title

@ 1,60 SAY DTOC(DATE()) + " " + TIME()

@ 2,0 SAY Uline

?

?

RETURN

* - - - - - Display error message.

PROCEDURE Error

PARAMETERS Message

@ 20,0

CLEAR

@ 20,3 SAY Message

? CHR(7)

WAIT "Press any key to try again"

RETURN

* - - - - - Look up customer by number or name

PROCEDURE GetCust

PARAMETERS M_Cust_No, M_Name, M_Address , Exiting

* - - Set up loop for validating customer number.

* - - Enter customer number or customer name and

* - - look form it.

Valid = .F.

DO WHILE .NOT. Valid

* - - - Get customer name or number

Lookup = SPACE(20)

@ 4,0 CLEAR

```
@ 15,5 SAY "Enter customer number or name"
GET Lookup
@ 17,5 SAY "Press Return to exit"
READ

* - - - If nothing entered , return "exiting"
IF Lookup = " "
    Exiting = .T.
    RETURN
ENDIF (Lookup is blank)

* Lookup by name if name entered.
IF ASC(Look up)>= 65
    Lookup = UPPER(TRIM(Lookup))
    SET INDEX TO CustName, CustNo
    SET EXACT OFF
    SEEK Lookup
    * - - - - - If name found
    IF FOUND()
        * - - - -Display customers with required
        * - - - - name
        M_Cust_No = Cust_No
        @ 5,0 CLEAR
        @ 6,0 SAY "Number   Name   Address"
        ?
        DISPLAY OFF WHILE UPPER(Cust_Name) = ;
        Lookup Cust_No , Cust_Name, Address
        @ 22,2 SAY "Enter customer number" GET ;
        M_Cust_No PICTURE "9999"
        READ
        Lookup = STR(M_Cust_No , 4)
    ELSE(If name not found)
        DO Error WITH "Not found"
    ENDIF(name not found)
ENDIF(name entered)
```



```

* - - - - - Lookup by customer number.
IF VAL(Lookup) > 0
  M_Cust_No = Val(Lookup)
  SET INDEX TO CustNo, CustName
  SEEK M_Cust_No
  * - - - - - If found, Continue, else ask again.
  IF FOUND()
    Valid = .T.
    M_Name = TRIM(Cust_Name)
    M_Address = TRIM(Address)
  ELSE
    DO Error with "Not found"
  ENDIF
ENDIF (number entered)
ENDDO (while invalid entry)
RETURN

```

١٩ - ٦ فتح ملف الخطوات

حتى يمكن تشغيل أى برنامج داخل ملف الخطوات يلزم فتح هذا الملف . ويتم ذلك باستخدام الأمر (SET PROCEDURE TO) ثم كتابة اسم الملف المطلوب . فمثلا لفتح ملف الخطوات الخاص ببرنامج حسابات العملاء (A/R) ، يتم كتابة السطر التالي :

```
SET PROCEDURE TO ProcLib1
```

وتنفيذ هذا الأمر يؤدي الى تحميل الملف (ProcLib1) في الذاكرة المؤقتة (RAM) ، وبالتالي يمكن تنفيذ أى برنامج داخل هذا الملف باستخدام الأمر (DO) كما سبق الايضاح .

ولاغلاق ملف الخطوات المفتوح يمكن كتابة الأمر (CLOSE PROCEDURE) . كما يمكن فتح ملف خطوات آخر . حيث أن هذا يؤدي الى اغلاق ملف الخطوات السابق . مع ملاحظة أنه لايمكن فتح أكثر من ملف خطوات في نفس الوقت .

١٩ - ٧ ادخال المعاملات (Parameters)

يستخدم الأمر (PARAMETERS) في ادخال المعاملات الى برامج الخطوات (Procedures) كما سبق الايضاح ، وهو يؤدي الى ادخال المعاملات التي يتم كتابتها مع الأمر (DO) في المتغيرات الموجودة بعده . ويتم ادخال المعاملات بنفس الترتيب . ويجب ملاحظة أن عدد المعاملات التي يتم ادخالها بعد الأمر (DO) يجب ان يكون مطابقا لعدد المتغيرات التي يتم كتابتها بعد الأمر (PARAMETERS) . والا فان البرنامج يتوقف ويعطي رسالة خطأ .

فمثلا السطور التالية تمثل انشاء برنامج خطوات اسمه (GetArea) يستخدم المتغيرات الثلاثة التالية (Length) ، (Width) ، (Area) .

```
PROCEDURE GetArea
PARAMETERS Length, Width, Area
Area = Length * Width
RETURN
```

ولتشغيل هذا البرنامج مع ادخال الطول (20) والعرض (25) والمساحة (Area) يتم كتابة الآتي :

```
Area = 0
DO GetArea WITH 20, 25, Area
```

وعند السؤال عن المساحة (Area) يتم كتابة الآتي :

```
? Area
```

ليلاحظ ظهور العدد (500) .

ويلاحظ هنا ضرورة تعريف المتغير (Area) قبل استخدامه . ويمكن استخدام أي متغيرات (X, Y, Z) كمعاملات مع تعريفها قبل استخدامها وذلك كالآتي :

```
X = 5
Y = 10
Z = 0
DO GetArea WITH X , Y, Z
```

وعند السؤال عن قيمة المتغير (Z) كالآتي :

? Z

يلاحظ ظهور العدد (50) .

ويجب ملاحظة أن الأمر (PARAMETERS) لا يتم استخدامه في ادخال المعاملات الى البرنامج فقط ولكنه يستخدم في اخراج قيم من البرنامج مثل المعامل (Area) مثلا . كما يجب ملاحظة أن ادخال المعاملات لا يستخدم فقط مع برامج الخطوات (Procedure) ولكنه يستخدم أيضا مع أى برنامج يتم انشاؤه بواسطة برامج عالية (DBase) .

١٩ - ٨ دراسة برنامج الخطوات (GetCust)

هذا البرنامج مكتوب في ملف الخطوات (ProcLib1) السابق شرحه . والبرنامج يبدأ بالسطين التاليين :

```
PROCEDURE GetCust
PARAMETERS M_Cust_No, M_Name, M_Address , Exiting
```

والمعاملات (Parameters) في هذا البرنامج تستخدم أساسا لاعادة البيانات المطلوبة الى البرنامج القائم بالاستدعاء .

والبرنامج يقوم بسؤال المستخدم عن الاسم أو الرقم المطلوب البحث عنه . ويقوم بتخزين هذا الاسم أو الرقم في المتغير (Lookup) ، وإذا لم يدخل المستخدم أى اسم فإن برنامج الخطوات يعود الى البرنامج المستدعى (Calling Program) مع اعادة القيمة (.T.) في المتغير (Exiting) . ويتم ذلك من خلال السطور التالية :

```
IF Lookup = " "
    Exiting = .T.
RETURN
ENDIF(lookup is blank)
```

وإذا أدخل المستخدم اسم عميل معين فإن البرنامج يحول هذا الاسم الى حروف كبيرة (Uppercase) ، ثم يبحث عن هذا الاسم . وذلك من خلال السطور التالية :

```

IF ASC(look up) >= 65
    Lookup = UPPER(TRIM(Lookup))
    SET INDEX TO CustName, CustNo
    SET EXACT OFF
    SEEK Lookup

```

ويتم استخدام الدالة (ASC) لمعرفة ما إذا كان المستخدم أدخل اسماً أم رقماً ، حيث أن كود الأسكني للحرف (A) هو (65) . أما الأرقام فإن الكود الخاص بها يكون أقل من (65) . ولذلك فإن تحقق الشرط بعد (IF) يعني أن المستخدم يدخل حرفاً وليس رقماً ، وهذا يعني أنه أدخل الاسم وليس رقم العميل . وفي هذه الحالة يتم فتح ملف الفهرس (CustName) باعتباره الفهرس الرئيسي ، ثم يتم البحث عن هذا الاسم . ويلاحظ هنا استخدام الأمر (SET EXACT OFF) للسماح للمستخدم بإدخال أي عدد من الحروف والحصول على كل الأسماء التي تبدأ بهذه الحروف .

وعند العثور على هذا الاسم يتم عرض بيانات جميع الأسماء المشتركة في هذا الاسم حتى يختار منهم المستخدم الاسم المطلوب . وفي هذه الحالة يقوم بإدخال رقم الحساب الخاص به ، وهذا يؤدي إلى الوصول إلى السجل المطلوب .

ويتم ذلك من خلال السطور التالية :

```

IF FOUND()
    * - - - - - Display customers with required
    * - - - - - name
    M_Cust_No = Cust_No
    @ 5,0 CLEAR
    @ 6,0 SAY "Number   Name   Address"
    ?
    DISPLAY OFF WHILE UPPER(Cust_Name) = ;
    Lookup Cust_No , Cust_Name, Address
    @ 22,2 SAY "Enter customer number" GET ;
    M_Cust_No PICTURE "9999"
    READ
    Lookup = STR(M_Cust_No , 4)

```

وفي حالة عدم العثور على هذا الاسم يتم عرض رسالة خطأ . ويلاحظ هنا أنه تم استخدام برنامج الخطوات (Error.prg) الموجود في نفس الملف . وهذا يعني أنه

يمكن استدعاء برنامج خطوات من برنامج خطوات آخر موجود داخل نفس ملف الخطوات (Procedure File) .

والسطور التالية توضح هذه العملية .

```
ELSE(If name not found)
  DO Error WITH "Not found"
ENDIF(name not found)
ENDIF(name entered)
```

والجزء التالي من البرنامج يعالج حالة ادخال المستخدم لرقم العميل دون اسمه .
وحيث أن القيم التي تنتج من الدالة (VAL) مع أى سلسلة حرفية تكون صفراً ، فإن الشرط الموجود بعد (IF) لا يتحقق إلا إذا كان ما يدخله المستخدم في المتغير (Lookup) أعداداً وليس حروفاً .

وحيث أن الحقل (Cust_No) هو حقل عددي لذلك يجب عند البحث عنه التأكد من أن القيمة الجارية مقارنتها قيمة عددية ، ولذلك تستخدم الدالة (VAL) لتحويل المتغير (Lookup) الى متغير عددي ، ثم يتم البحث عن هذا المتغير بعد فتح الفهرس الخاص برقم العميل (CustNo) . ويتم ذلك من خلال السطور التالية :

```
IF VAL(Lookup) > 0
  M_Cust_No = Val(Lookup)
  SET INDEX TO CustNo, CustName
  SEEK M_Cust_No
```

وإذا تم العثور على رقم العميل يتم تغيير المتغير (Valid) الى (.T.) وهذا يؤدي الى عدم تنفيذ الحلقة التكرارية مرة ثانية ، ثم يتم تخزين الاسم (Cust_Name) الخاص بالسجل الذي تم الوصول اليه في المتغير (M_Name) والعنوان الخاص به (Address) في المتغير (M_Address) . وإذا لم يتم العثور على رقم العميل يتم عرض رسالة خطأ (Error Message) . والسطور التالية توضح هذه العملية :

```
IF FOUND()
  Valid = .T.
  M_Name = TRIM(Cust_Name)
  M_Address = TRIM(Address)
ELSE
```

```
DO Error with "Not found"
ENDIF(not eof)
ENDIF(number entered)
ENDDO(While invalid entry)
RETURN
```

ويلاحظ خلال هذا البرنامج أن ملف قاعدة البيانات لم يتم فتحه وذلك لأنه يكون قد تم فتحه من خلال البرنامج الرئيسي لنظام حسابات العملاء . كما يجب ملاحظة أن ملف الخطوات يجب فتحه من خلال البرنامج الرئيسي حتى يمكن استخدام البرامج الموجودة فيه في البرامج الفرعية للنظام .

الفصل العشرون

برامج القائمة الرئيسية والادخال والتعديل

في هذا الفصل سوف يتم دراسة برنامج القائمة الرئيسية لحسابات العملاء (Main Menu) وبرنامج ادخال عملاء جدد وبرنامج ادخال حركة الصرف وبرنامج ادخال حركة السداد بالإضافة الى برامج التعديل . وحيث أن هذه البرامج لا تختلف من برامج المخازن السابق شرحها من حيث تصميمها المنطقي ، لذلك فسوف يتم كتابتها دون شرحها مع شرح أى أوامر جديدة لم يسبق استخدامها .

٢٠ - ١ برنامج القائمة الرئيسية

هذا البرنامج لا يختلف عن باقى برامج القائمة الرئيسية . حيث أنه يعرض قائمة على المستخدم للاختيار منها . وبناء على اختيار المستخدم يقوم البرنامج بالتفرع الى البرامج الأخرى . ولكن فى هذا البرنامج يتم كتابة الأمر التالى :

SET PROCEDURE TO ProcLib1

وذلك لفتح ملف الخطوات (ProcLib1) السابق شرحه فى الفصل السابق . كما يجب كتابة الأمر (CLOSE PROCEDURE) قبل نهاية البرنامج .

وعند تشغيل هذا البرنامج تظهر الشاشة الموضحة بالشكل (٢٠ - ١) .

02/20/90	09:50:30
Accounts Receivable Main Menu	
<ol style="list-style-type: none"> 1. Add new customers 2. Enter new charges 3. Enter payments/Adjustments 4. Print reports 5. Edit data 6. Do monthly posting 7. Exit 	
Enter choice (1 - 7)	

شكل (٢٠ - ١)

ويتكون هذا البرنامج من السطور التالية :

```
*****AR.prg
*   Accounts Receivable Main menu
CLEAR ALL

* - - - - Declare variables as public for passing to
* - - - - procedures.
PUBLIC M_Cust_No, M_Name, M_Address, Message, Exiting

* - - - - - Open Procedure file ProcLib1.prg
SET PROCEDURE TO ProcLib1

* - - - - - If color monitor in use , set colors .
IF ISCOLOR()
    SET COLOR TO GR+/B , W+/RB
ENDIF

* - - - - - Set Parameters
SET BELL OFF
SET DELETED ON
SET DEVICE TO SCREEN
SET HEADING OFF
SET STATUS OFF
SET TALK OFF

* - - - - Get the date (RUN needs about 320KB RAM).
?
RUN DATE

* - - - - - Create underline variable Uline.
Uline = REPLICATE("_" , 80)

* - - - - - Set up a loop for the main menu.
Choice = 0
DO WHILE Choice # 7
```

```
CLEAR
* - - - - - Print screen title
DO Title WITH "Accounts Receivable Main Menu"
TEXT
    1. Add new customers
    2. Enter new charges
    3. Enter payments / adjustments
    4. Print reports
    5. Edit data
    6. Do monthly posting
    7. Exit
ENDTEXT
@ 23,1 SAY "Enter choice (1 - 7) " ;
GET Choice PICT "9" RANGE 1,7
READ
* - - - - - Branch accordingly.
DO CASE
    CASE Choice = 1
        DO NewCust
    CASE Choice = 2
        DO NewChrg
    CASE Choice = 3
        DO NewPay
    CASE Choice = 4
        DO ARPrint
    CASE Choice = 5
        DO AREdit
    CASE Choice = 6
        DO Post
ENDCASE
ENDDO(while choice # 7)

* - - - - - Close procedure file and exit.
CLOSE PROCEDURE
CLEAR
QUIT
```

ويلاحظ في هذا البرنامج استخدام الأمر (PUBLIC) مع المتغيرات (M_Cust No) ، (M_Name) ، (M_Address) ، (Exiting) . وذلك حتى يمكن استخدام هذه المتغيرات في جميع البرامج الفرعية دون الحاجة الى اعادة انشائها من جديد . كما أن هذا يعتبر ضروريا لاستخدام برنامج الخطوات (GetCust) . وحتى لا نحتاج الى اعادة انشاء هذه المتغيرات في كل مرة يراد فيها استخدام هذا البرنامج .

ويلاحظ أيضا استخدام برنامج الخطوات (Title) لكتابة عنوان الشاشة ورسم خط فيها .

ويلاحظ أيضا استخدام الأمر (SET COLOR) في تجميل الشاشة وزيادة جاذبيتها ، وهذا الأمر مع المعاملات الموجودة معه يؤدي الى تكوين خلفية زرقاء مع الكتابة باللون الأصفر في الشاشة الرئيسية ، كما يؤدي الى تكوين خلفية بنفسجية مع الكتابة باللون الأبيض في الأعمدة الضوئية (Highlights) . ويمكن تعديل هذه الألوان حسب الحاجة .

والأمر (RUN DATE) تم استخدامه لتشغيل برنامج التاريخ (DATE) الموجود في نظام التشغيل (MS-DOS) حتى يسمح للمستخدم بتعديل التاريخ الحالي حسب الحاجة . ويجب ملاحظة أن استخدام الأمر (RUN) من خلال (DBase III+) يتطلب ذاكرة مؤقتة لا تقل عن ٢٢٠ ك بايت .

٢٠ - ٢ برنامج اضافة العملاء (NewCust.prg)

هذا البرنامج لا يختلف كثيرا عن برنامج اضافة أرقام الأجزاء الجديدة في نظام المخازن السابق شرحه ولكن نظرا لأن نظام حسابات العملاء يعتمد على ربط الملفات، لذلك فإن هذا البرنامج يضيف جزءا جديدا لاختبار كل رقم عميل جديد يتم ادخاله والتأكد من عدم تكراره . كما أن البرنامج يقوم بزيادة رقم العميل أكيا مع كل اضافة لعميل جديد ويسمح للمستخدم بقبول هذا الرقم أو ادخال رقم آخر جديد .

وعندما يختار المستخدم الرقم (1) من القائمة الرئيسية للبرنامج الرئيسي (AR.prg) ، فإن البرنامج يمسح الشاشة ويعرض الرسالة التالية :

Enter Customer number (0 to quit) : 1001

في هذه الحالة يستطيع المستخدم الضغط على مفتاح الادخال للموافقة على هذا الرقم أو كتابة صفر (0) للرجوع الى القائمة الرئيسية ، أو كتابة رقم حساب آخر .

وعندما يكتب المستخدم رقما موجودا تظهر الرسالة التالية :

Number already in use!

ثم يسمح البرنامج للمستخدم بالمحاولة مرة ثانية .

وعندما يكتب المستخدم رقما غير موجود تظهر شاشة الادخال حتى يستطيع المستخدم ادخال باقي بيانات هذا العميل . والشكل (٢٠ - ٢) يوضح شاشة الادخال المستخدمة في هذا البرنامج ، والتي يتم انشاؤها من خلال قوائم برنامج المساعد (Assistant) او عن طريق كتابة السطرين التاليين من مشيرة النقطة . (Dot Prompt)

<u>Add New Customers</u>	
Customer Number :	9999
Customer Name :	XXXXXXXXXXXXXX
Address :	XXXXXXXXXXXXXXXXXXXXX
Phone :	XXXXXX
<hr/>	
Starting Balance :	99999.99
Terms :	XXXXXXXXXXXXX
Balance(30 days):	99999.99
Balance(60 days):	99999.99
Balance(90 days):	99999.99
Balance(90 + days):	99999.99

شكل (٢٠ - ٢)

USE Customer
CREATE SCREEN FNewCust

وفي هذه الحالة تظهر قوائم الاختيارات التي يتم عن طريقها تحميل الحقول

المطلوب ظهورها على السبورة (Blackboard) . ثم يتم تحديد أماكن هذه الحقول على الشاشة كما سبق الايضاح في الكتاب الأول .

وحيث أن رقم العميل يتم التحكم فيه من خلال البرنامج ، لذلك يجب عدم السماح للمستخدم بتعديله . وهذا يتم تنفيذه عند تصميم الشاشة . حيث يتم تحريك مؤشر التصحيح إلى العمود الضوئي المقابل لرقم العميل (Customer Number) ثم الضغط على مفتاح (F10) ثم الضغط على مفتاح الإدخال عند وقوف المؤشر على الاختيار (ACTION) لتحويله إلى (Display/Say) . ثم يتم الضغط على مفتاح (F10) مرة أخرى للرجوع إلى السبورة (Blackboard) . ويلاحظ في هذه الحالة اختفاء العمود الضوئي (Highlight) مع بقاء الأرقام (9999) الدالة على نوع الحقل . وهذا يعني أن المستخدم يرى رقم العميل ولكن لا يمكنه تعديله .

والبرنامج المستخدم في إضافة العملاء يسمى (NewCust.prg) ويتكون من السطور التالية :

```
***** NewCust.prg
*   Add new customers to the A/R system.
*   Called from AR main menu
USE Customer INDEX CustNo, CustName

* - - Store largest customer number to memory variable
* - - Next_No.
GO BOTT
Next_No = Cust_No

* - - - - Print screen title.
DO Title WITH "Add New Customers"

* - - - - Set up loop for adding customers.
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - - - Increment customer number by 1
    Next_No = Next_No + 1
    * - - Suggest next number , but allow user to
    * - - change it.
    @ 15,5 SAY "Enter customer number (0 to quit): " ;
    GET Next_No PICT "9999"
```

```

@ 17,5 SAY "Press Return to accept number "
READ
* - - - - If zero entered , return to main menu.
IF Next_No = 0
    Exiting = .T.
    LOOP
ENDIF

* - - - Check to see if number is already taken.
* - - - If it is, loop and ask for another number.
SEEK Next_No
IF FOUND()
    ? "Number already in use" , CHR(7)
    LOOP
ENDIF

* - -If next number isn't taken , add new record.
* - -using FNewCust format screen.
APPEND BLANK
REPLACE Cust_No WITH Next_No
REPLACE Term WITH "Net30"
REPLACE Last_Updat WITH DATE()
SET FORMAT TO FNewCust
READ
SET FORMAT TO
@ 4,0 CLEAR
ENDDO (While not exiting)
* - - - - - End of program
CLOSE DATABASES
RETURN
    
```

ويلاحظ في هذا البرنامج استخدام المتغير (Next_No) في زيادة رقم السجل
 بواحد ألياً وكذلك في اختبار رقم العميل والتأكد من عدم وجوده في ملف العملاء
 . (Customer.dbf)

٢٠ - ٢ برنامج اضافة حركة الصرف (Newchrg.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (2) من القائمة الرئيسية للبرنامج (AR.prg) .

يستخدم هذا البرنامج في اضافة بيانات حركة الصرف الى ملف قاعدة البيانات الخاص بحركة الصرف للعملاء (Charges.dbf) . ولكتابة هذا البرنامج يلزم أولا انشاء شاشة تعديل بيانات الصرف (FNewChrg.fmt) وذلك بكتابة السطرين التاليين من مشيرة النقطة .

```
USE Charges
CREATE SCREEN FNewChrg
```

ويتم تحميل كل حقول ملف قاعدة البيانات (Charges.dbf) ما عدا حقلي الكمية (Amount) والدفع (Billed) . وكما حدث في شاشة ادخال العملاء (FNewCust) يتم حماية حقول رقم العميل من التعديل . وذلك باستخدام المفتاح (F10) وتحويل الاختيار (Action) الى (Display/Say) .

وعند الانتهاء من تصميم الشاشة تصبح بالصورة الموضحة بالشكل (٢٠ - ٢) . وهنا يجب ملاحظة أن اسم العميل وعنوانه لم يتم اضافته الى الشاشة . حيث أنه لا يكون ضمن حقول ملف حركة الصرف (Charges.dbf) . ولكن في العادة يحتاج المستخدم الى معرفة اسم العميل وعنوانه أثناء تعديل البيانات الخاصة به . لذلك يتم اضافة متغيرات الذاكرة (M_Name) ، (M_Address) الى شاشة الادخال . ولتنفيذ ذلك يتم استخدام الأمر (MODIFY COMMAND) في تعديل برنامج تصميم الشاشة وذلك كالآتي :

```
MODIFY COMMAND FNewchrg.fmt
```

ثم يتم كتابة السطرين التاليين :

```
@ 4,33 SAY "Name:" + M_Name
@ 5,33 SAY "Address:" + M_Address
```

مع ملاحظة أن الاحداثيات يمكن تعديلها بناء على شكل الشاشة الذي سبق تصميمه .

Add New Payments	
Customer Number :	9999
Check Number :	9999999
Date :	99/99/99
Part Number :	99999
Qty :	99999 99
Unit Price:	99999 99
Description :	xxxxxxxxxxxxxxxxxxxxxxxxxxxx
Cursor Movement by Up, Down, Left, and Right arrow keys Insert Mode: Ins Delete Characters: Del Save: End or ~ W Abandon: ~ Q	

شكل (٢٠ - ٢)

وحتى يختبر مخطط البرامج الشاشة بعد ادخال هذا التعديل يمكنه كتابة الأوامر التالية من مشيرة النقطة (Dot Prompt).

```
M_name = "Mohamed"
M_Address = "12 Hegaz street"
USE Charges
SET FORMAT TO FNewChrg
APPEND
```

فاذا ظهرت الشاشة فان هذا يدل على أن ملف التشكيل (Format File) سليم . واذا لم تظهر فان هذا يدل على وجود خطأ معين في ملف التشكيل . ويلاحظ أنه تم انشاء متغيرات الذاكرة (M_Name) ، (M_Address) قبل اجراء الاختبار وذلك لأن هذه المتغيرات ليست موجودة ضمن حقول ملف حركة المصرف (Charges.dbf) . أما أثناء تنفيذ برنامج الاضافة (NewChrg.prg) فان هذه المتغيرات يتم انشاؤها من خلال برنامج الخطوات (GetCust.prg) الذي سبق شرحه .

ويتم كتابة سطور برنامج الاضافة كالآتي :

```

*****NewChrg.prg.
* Add individual charges to the Charges database file.
* Called from AR main menu

* - - - - - Print the screen title
DO Title WITH "Enter New Charges"
* - - - - - Open both Charges and Customer databases.
SELECT 1
USE Customer INDEX CustNo , CustName
SELECT 2
USE Charges INDEX ChrgNo

* - - - - - Set up loop for adding entries.
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - - Get customer name or number , and validate.
    SELECT 1
    DO GetCust WITH ;
        M_Cust_No, M_Name, M_Address,Exiting
    * - - - Allow user to enter transaction data using
    * - - - FNewChrg screen.
    IF .NOT. Exiting
        SELECT 2
        APPEND BLANK
        REPLACE Cust_No WITH M_Cust_No
        REPLACE Date WITH DATE()
        REPLACE Billed WITH .F.
        SET FORMAT TO FNewChrg
        HEAD
        CLOSE FORMAT
        REPLACE Amount WITH Qty * Unit_Price
    ENDIF
    @ 3,0 CLEAR
ENDDO(While adding new transactions)
* - - - - - Return to main menu.
CLOSE DATABASES
RETURN

```

ملاحظة

إذا أريد إضافة الضريبة الى الكمية (Amount) يمكن إضافة السطر التالي مثلا :

$Amount = 1,05 * Amount$

وذلك قبل انتهاء الحلقة التكرارية .

٢٠ - ٤ برنامج إضافة حركة السداد (NewPay.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (٢) من القائمة الرئيسية للبرنامج (AR.prg) . ويستخدم هذا البرنامج لى إضافة بيانات حركة السداد (Payments) الى ملف قاعدة البيانات الخاص بحركة السداد (Payments.dbf) .

وهذا البرنامج يماثل تماما برنامج إضافة حركة الصرف . حيث يتم انشاء شاشة ادخال بنفس الطريقة عن طريق كتابة السطرين التاليين :

```
USE Payments
CREATE SCREEN FNewPay
```

وفى هذه الحالة تظهر قوائم الاختيارات الخاصة برسم الشاشة . ويتم تحميل حقول ملف حركة السداد (Payments.dbf) كلها باستثناء حقل الترحيل (Posted) . كما يتم استخدام مفتاح (F10) لتحويل حقل رقم العميل (Cust_No) الى حالة (Display/Say) حتى يصبح محميا من أى تعديل بواسطة المستخدم .

ولإضافة المتغيرين (M_Name) ، (M_Address) الى شاشة الادخال يتم استخدام نفس الطريقة السابقة . وذلك باستخدام الأمر (MODIFY COMMAND) عن طريق كتابة السطر التالى :

```
MODIFY COMMAND FNewPay.FMT
```

ثم يتم إضافة السطرين التاليين الى برنامج رسم الشاشة .

```
@ 5,35 SAY "Name:" + M_Name
@ 6,35 SAY "Address:" + M_Address
```

وعند الانتهاء من تصميم الشاشة تصبح بالصورة الموضحة بالشكل (٢٠ - ٤) .

Add New Payments	
Customer Number : 9999	Name :
Check Number : 9999999	Address :
Amount : 9999999 99	
Description : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
Cursor Movement by Up, Down, Left, and Right arrow keys.	
Insert Mode : Ins	Delete Character : Del
Save : End or ~ W	Abandon : ~ Q

شكل (٢٠ - ٤)

ويمكن استخدام هذه الشاشة في اضافة أي حركة سداد (Payments) . كما يمكن اضافة أي حركة ضبط (adjustment) عند ارجاع العميل لأي صنف ويراد اضافته للرصيد . وفي هذه الحالة يتم ملء بيانات الشاشة مع تحديد الكمية التي تمت اعاتدها . ثم يتم توضيح سبب رجوع هذه الكمية في حقل الوصف (Discript) .

وكما تم بالنسبة لبرنامج اضافة حركة الصرف ، يتم استخدام برنامج الخطوات (GetCust) في البحث عن اسم العميل أو رقمه والتأكد من وجوده في الملف . وكما سوف نلاحظ فان البرنامج يعاقل برنامج اضافة حركة الصرف (NewChrg.prg) مع اختلافات قليلة جدا . لذلك يمكن نسخه باستخدام الأمر (MODIFY COMMAND) وقراءة نفس هذا الملف في الملف الجديد . وذلك بكتابة (*KR) ثم اجراء التعديلات المطلوبة فيه . وذلك بدلا من كتابته من جديد . والسطور التالية توضح أوامر هذا البرنامج .

```

* * * * * NewPay.prg.
*   Add individual payments to the payments file.
*   Called from AR main menu

* - -Open both Payments and Customer databases.
SELECT 1
USE Customer INDEX CustNo , CustName
SELECT 2
USE Payments INDEX PayNo

* - - - - - Set up loop for adding entries.
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - Get customer name or number , and validate.
    SELECT 1
    DO GetCust WITH ;
        M_Cust_No , M_Name , M_Address , Exiting
    * - Allow user to enter transaction data using
    * - FNewPay screen.
    IF .NOT. Exiting
        SELECT 2
        APPEND BLANK
        REPLACE Cust_No WITH M_Cust_No
        REPLACE Date WITH DATE()
        REPLACE Discript WITH "Payment"
        REPLACE Posted WIHT .F.
        SET FORMAT TO FNewPay
        READ
        CLOSE FORMAT
    ENDIF
    @ 3,0 CLEAR
ENDDO (While adding new transactions)
* - - - - - Return to main menu.
CLOSE DATABASES
RETURN

```

٢٠ - ٥ برنامج التعديل (AREdit.prg)

يتم تعديل حسابات العملاء باستخدام برنامج تعديل اضافة العملاء (EdCust.prg) ، وبرنامج تعديل حركة الصرف (EdChrg.prg) ، وبرنامج تعديل حركة السداد (EdPay.prg) . ويتم التحكم في هذه البرامج من طريق برنامج قائمة التعديل (AREdit.prg) . ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (5) من القائمة الرئيسية للبرنامج (AR.prg) . وعند تشغيل هذا البرنامج تظهر القائمة الموضحة بالشكل (٢٠ - ٥) .

02/20/90	09:50:40
Accounts Receivable Edit Menu	
<ol style="list-style-type: none">1. Edit Customer File2. Edit Current Charges3. Edit Current Payments4. Return to Main Menu	
Enter choice (1 - 4) <input type="text"/>	

شكل (٢٠ - ٥)

كما يتم كتابة هذا البرنامج كالآتي :

```

* * * * *AREdit.prg
*   Menu for editing the A/R System.
*   Called from AR main menu
EChoice = 0
DO WHILE EChoice # 4
    CLEAR
    DO Title WITH "Accounts Receivable Edit Menu"
    TEXT
        1. Edit Customer File
        2. Edit Current Charges
        3. Edit Current Payments
        4. Return to main menu
    ENDTEXT
    @ 24,3 SAY "Enter choice (1 - 4) " ;
    GET EChoice PICT "9" RANGE 1,4
    READ

    * - - - - Branch accordingly.
    DO CASE
        CASE EChoice = 1
            DO EdCust
        CASE EChoice = 2
            DO EdChrg
        CASE EChoice = 3
            DO EdPay
    ENDCASE
ENDDO (Echoice # 4)
* - - - - Return to main menu
RETURN

```

٢٠ - ٥ - ١ تعديل ملف العميل (EdCust.prg)

شاشة تعديل ملف العميل (FEEdCust.scr) هي نفس شاشة ادخال العملاء (FNewCust.scr) مع اختلاف وحيد وهو أن العنوان يجب تغييره الى (Edit Customers) . ولذلك يتم نسخ ملف شاشة العميل في الملف الجديد (FEEdCust.scr) وذلك كالآتي :

COPY FILE FNewCust.scr TO FEEdCust.scr

ثم يتم كتابة السطر التالي :

MODIFY SCREEN FEEdCust

في هذه الحالة تظهر السبورة (Blackboard) ويتم تعديل العنوان من (Add New Customers) الى العنوان (Edit Customers) ، ثم تخزين الشاشة الجديدة . أنظر الشكل (٢٠ - ٦) .

<i>Edit Customers</i>	
Customer Number :	9999
Customer Name :	XXXXXXXXXXXXXXXXXX
Address :	XXXXXXXXXXXXXXXXXXXX Phone : XXXXXXXX
Starting Balance : 99999.99 Terms: XXXXXXXXXXXXX	
Balance(30 days):	99999.99 Balance(60 days): 99999.99
Balance(90 days):	99999.99 Balance(90 + days): 99999.99

شكل (٢٠ - ٦)

وبرنامج تعديل ملف العميل يسمى (EdCust.prg) وهو يستخدم برنامج الخطوات (GetCust.prg) في البحث عن العميل برقمه أو باسمه مثل البرامج السابقة . ويتم كتابة سطور البرنامج كالآتي :


```

***** EdCust.prg
*   Edit Customers information
*   Called from A/R Edit menu
* - - - - -Print the screen title.
DO Title WITH "Enter Customers File"
USE Customer INDEX CustNo, CustName
* - - - - - Set up loop for editing.
Exiting = .F.
SET DELETED OFF
DO WHILE .NOT. Exiting
    * - - - - - GET Customer by name or number
    DO GetCust WITH ;
        M_Cust_No , M_Name , M_Address , Exiting
    * - Edit using the FEdCust screen (if not exiting)
    IF .NOT. Exiting
        SEEK M_Cust_No
        SET FORMAT TO FEdCust
        EDIT
        SET FORMAT TO
        @ 4,0 CLEAR
    ENDIF
ENDDO (While not exiting)
* - - - - - Return to main menu
SET DELETED ON
CLOSE DATABASES
RETURN

```

٢-٥-٢٠ تعديل ملف الصرف (EdChrg.prg)

برنامج تعديل ملف الصرف (Charges.dbf) يسمح للمستخدم بإدخال رقم العميل أو اسمه لتحديد مكان هذا العميل في الملف ، ثم استخدام مفتاحي (PgDn) ، (PgUp) في عرض كل حركة خاصة بهذا العميل للوصول الى الحركة المطلوب تعديلها . وتستخدم الشاشة (FEdChrg.scr) في تعديل حركة الصرف . ولانشاء هذه الشاشة يتم نسخ شاشة اضافة الصرف (FNewchrg.scr) في شاشة التعديل (FEdchrg.scr) وذلك بكتابة السطر التالي :

COPY FILE FNewChrg.scr TO FEdChrg.scr

ثم يتم كتابة السطر التالى :

MODIFY SCREEN FEdChrg

حتى يتم عرض السبورة (Blackboard) وتعديل عنوان الشاشة الى (Edit Charges) كما يتم اضافة تعليمات أسفل الشاشة لتوضيح للمستخدم كيفية الانتقال من حركة الى حركة أخرى باستخدام مفتاحي (PgDn) ، (PgUp) .

وحتى يظهر اسم العميل وعنوانه مع كل حركة يلزم فتح ملف العملاء (Customer.dbf) ، ويطه بملف المصرف ، وتثبيت مؤشر السجلات (Record Pointer) على السجل الخاص بهذا العميل ثم الحصول على اسم هذا العميل وعنوانه . ويتم ذلك من طريق تعديل ملف الشاشة بواسطة الأمر :

MODIFY COMMAND FEdchrg.fmt

ثم يتم كتابة السطرين التاليين فى هذا الملف

```
@ 4,33 SAY "Name:" + TRIM(A-> Cust_Name)
@ 5,33 SAY "Address:" + TRIM(A-> Address)
```

ويجب ملاحظة أن الاحداثيات الخاصة بهذين السطرين تتوقف على تصميم الشاشة . والشكل (٢٠ - ٧) يوضح صورة الشاشة المستخدمة فى تعديل حركة المصرف .

ولكى يتم التحكم فى مؤشر السجلات (Record Pointer) الموجود فى ملف العملاء للحصول على اسم العميل وعنوانه يجب انشاء علاقة بين ملفي العملاء (Cust.dbf) والمصرف (Charges.dbf) وذلك من خلال برنامج تعديل ملف المصرف (EdChrg.prg) ويتضح ذلك من السطور التالية :

<i><u>Edit Charges</u></i>	
Customer Number : 9999	Name :
Invoice Number : 9999999	Address :
Part Number : 99999 Qty : 99999.99	Date : 99/99/99
Unit Price: 99999.99	
Description : xxxxxxxxxxxxxxxxxxxxxxxxxxxx	
Cursor Movement by Up, Down, Left, and Right arrow keys:	
Insert Mode: Ins	Delete Characters: Del
Save: *End or *W	Attention: *O

شكل (٢٠ - ٧)

* - - - - - Open both Charges and Customer databases.

SELECT 1

USE Customer INDEX CustNo, CustName

SELECT 2

USE Charges INDEX ChrgNo

* - - - - - Set up relationship

SET RELATION TO Cust_No INTO Customer

وهذا الربط يجعل المؤشر يقف دائماً على نفس السجل الخاص بالعميل بناءً على رقم العميل بصرف النظر عن الحركة التي يتم عرضها على الشاشة . وبالتالي يظل اسم العميل وعنوانه معروضاً على الشاشة أثناء تعديل أي حركة خاصة بهذا العميل .

والبرنامج يتم كتابته سطره كالاتي :

```

* * * * * EdChrg.prg
*   Edit invalid charges in the charges file.
*   Called from A/R Edit menu

* - -Print the screen title.
DO Title WITH "Edit Current Charges"
* - - - - - Open both Customer and charges databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Charges INDEX ChrgNo
* - - - - - Set up relationship
SET RELATION TO Cust_No INTO Customer
* - - - Set up memory variables and loop for editing
SET DELETED OFF
Exiting = .F.
DO WHILE .NOT. Exiting
* - - - - - Get Customer by name or number .
SELECT 1
DO GetCust WITH M_Cust_No, M_Name, M_Address, Exiting
* - Edit the transaction, if valid and not exiting
IF .NOT. Exiting
    SELECT 2
    SEEK M_Cust_No
    IF FOUND .AND. Billed
        SET FORMAT TO FedChrg
        EDIT
        CLOSE FORMAT
    ELSE
        DO Error WITH "Already posted,Make adjustment"
    ENDIF
ENDIF (not exiting)
@ 4,0 CLEAR
ENDDO (While not exiting)
* - - - - - Recalculate amount field.
SELECT 2
REPLACE ALL Amount WITH Qty * Unit_Price
SET DELETED ON

```

CLOSE DATABASES

* - - - - Return to main menu.

RETURN

ويلاحظ من البرنامج استخدام الجملة

IF FOUND() .AND. .NOT. Billed

حيث أنها تمنع تعديل السجل بعد ترحيله الى الملف الرئيسي . أما اذا كان قد تم ترحيله فتظهر رسالة خطأ (Error Message) توجه المستخدم الى التعديل عن طريق اضافة حركة جديدة الى ملف حركة الصرف ، كما يلاحظ بعد انتهاء عملية التعديل اضافة السطر التالي :

REPLACE ALL Amount WITH Qty * Unit_Price

وهذا لضمان أن أى تعديل في الكمية المصروفة أو في سعرها يتم ادخاله مباشرة في كمية الصرف الكلية لهذا المعيل .

٢٠ - ٥ - ٣ تعديل ملف السداد (EdPay.prg)

هذا البرنامج يماثل تعديل ملف الصرف تماما مع بعض الاختلافات البسيطة ، وهو يستخدم الشاشة (FedPay.scr) في التعديل ، ويتم انشاؤها بنسخ الشاشة (FNewPay.scr) وذلك كالآتي :

COPY FILE FNewPay.scr TO FedPay.scr

ثم يتم استخدام الأمر (MODIFY SCREEN) في تحويل العنوان الى (Edit Payments) و اضافة الارشادات الخاصة باستخدام مفتاحي (PgUp) ، (PgDn) في الانتقال من حركة الى أخرى .

كما يتم استخدام الأمر (MODIFY COMMAND Fedpay.fmt) في ادخال اسم المعيل وعنوانه في الشاشة . وذلك عن طريق كتابة السطرين التاليين في ملف الشاشة (Fedpay.fmt) :

@ 5,35 SAY "Name :" + TRIM(A -> Cust_Name)

@ 6,35 SAY "Address:" + TRIM(A -> Address)

والشكل (٢٠ - ٨) يوضح صورة شاشة الادخال .

<i>Edit Payments</i>	
Customer Number : 9999	Name :
Check Number : 9999999	Address :
Amount : 9999999.99	Date : 99/99/99
Description : XXXXXXXXXXXXXXXXXXXXXXXX	
Cursor Movement by Up, Down, Left, and Right arrow keys.	
Insert Mode : Ins	Delete Character : Del
Save : End or W	Attention : O

شكل (٢٠ - ٨)

وبرنامج التعديل يتكون من السطور التالية :

```

.....EdPay.prg
*   Edit invalid payments in the payments file.
*   Called from AR Edit menu
* - - - - - Print the screen title.
DO Title WITH "Edit Current Payments"
* - - - - - Open both Customer and payments databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Payments INDEX PayNo

* - - - - - Set up relationship
SET RELATION TO Cust_No INTO Customer
    
```

```
* - - - - - Set up loop for editing entries
SET DELETED OFF
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - - - - Get Customer by name or number
    SELECT 1
    DO GetCust WITH ;
        M_Cust_No, M_Name, M_Address, Exiting
    * - - If not exiting , and transaction not already
    * - - - - - posted , proceed with edit.
    IF .NOT. Exiting
        SELECT 2
        SEEK M_Cust_No
        IF FOUND() .AND. .NOT. Posted
            SET FORMAT TO FedPay
            EDIT
            CLOSE FORMAT
        ELSE
            DO Error WITH ;
                "Already posted, Make adjustment:"
        ENDIF
    ENDIF (not exiting)
    @ 4,0 CLEAR
ENDDO(while not exiting)
* - - - - - Close databases and return to edit menu
SET DELETED ON
CLOSE DATABASES
RETURN
```


الفصل الحادي والعشرون

تقارير برنامج حسابات العملاء

يعتمد برنامج حسابات العملاء (A/R) على مجموعة من التقارير (Reports) . مثل الفواتير (Invoices) والتقارير الشهرية والتقارير المختصرة (Summary reports) والتقارير الزمنية (Aging reports) التي توفر للمستخدم المتابعة الدقيقة للبيانات وتقديم الخدمة السريعة للعميل . وكذلك التقارير التاريخية (Historical Reports) التي تساعد على اتخاذ القرارات وأجراء التعديلات المطلوبة في النظام بناء على دراسات احصائية للمخرجات .

وفي الواقع فان كتابة البرامج التي توفر هذه الأنواع من التقارير تحتاج الى كثير من الوسائل المتقدمة في كتابة البرامج وخصوصا عندما يراد الاحتفاظ بسرعة تشغيل البرنامج وكفاءته . ولذلك فسوف يتم استخدام برامج الخطوات (Procedures) والمعاملات (Parameters) في بعض البرامج كما سيتم تصميم برامج لأنواع مختلفة من التقارير .

٢١ - ١ برنامج قائمة التقارير الرئيسية (ARPrint.prg)

عندما يختار المستخدم الرقم (4) من القائمة الرئيسية لبرنامج حسابات العملاء (A/R) تظهر شاشة الاختيارات الموضحة بالشكل (٢١ - ١) .

A/R Print Menu

- 1- Print monthly statements
- 2- Print monthly summary
- 3- Print aging reports
- 4- Do quick looking of customer status
- 5- Review history
- 6- Return to main menu

Enter choice (1 - 6)

شكل (٢١ - ١)

والاختيار الأول يستخدم عندما يراد طباعة الفواتير (Invoices). ويجب تشغيله مرة واحدة في الشهر عندما يراد ترحيل الحسابات إلى الملف الرئيسي .

والاختياران (2 , 3) يستخدمان أيضا مرة واحدة في الشهر بعد طباعة الفواتير للحصول على تقرير مختصر عن موقف الفواتير الخاصة بكل عميل .

والاختيار رقم (4) يستخدم عندما يريد العميل مراجعة الحساب الخاص به .

والاختيار رقم (5) يوفر مراجعة أي بيانات تاريخية لأي حساب لمساعدة المستخدم على متابعة المسير التاريخي لأي عميل ، وتوفير المعلومات التي تتعلق بفترة معينة محصورة بين تاريخين ، وكذلك توفير معلومات عن حالة بيع صنف معين .

والبرنامج الخاص بعرض القائمة الرئيسية للتقارير (ARPrint.prg) لا يختلف عن أي برنامج من برامج القائمة الرئيسية الأخرى . ويتكون من السطور التالية :

```
*****ARPrint.prg
*   Menu of Print options for the A/R system.
*   Called from A/R main menu

Pchoice = 0
SET DELETED ON
DO WHILE Pchoice # 6
    CLEAR
    DO Title WITH "A/R Print Menu"
    TEXT
        1. Print monthly statements
        2. Print monthly Summary
        3. Print aging reports
        4. Do quick looking of customer status
        5. Review history
        6. Return to main menu
    ENDTEXT
    @ 24,1 SAY "Enter choice (1 - 6)" ;
    GET Pchoice PICT "9" RANGE 1,6
    READ
* - - - - - Branch accordingly.
```

```

DO CASE
  CASE PChoice = 1
    DO Bills
  CASE PChoice = 2
    RepForm = "ARSumm"
    DO AgeSumm
  CASE PChoice = 3
    RepForm = "Aging"
    DO AgeSumm
  CASE PChoice = 4
    DO ARStat
  CASE PChoice = 5
    DO ARHist
ENDCASE
ENDDO(PChoice # 6)
* - - - - - Return to main menu
SET DELETED OFF
RETURN

```

ويلاحظ من هذا البرنامج أن الاختيار (2) والاختيار (3) يؤديان إلى تنفيذ نفس البرنامج (AgeSumm) ولكن البرنامج يعطي تقريراً مختلفاً في كل حالة كما سيتم الإيضاح فيما بعد .

٢١ - ٢ ملف الخطوات (BillProc.prg)

يلاحظ من البرنامج السابق الخاص بقائمة التقارير وجود برنامج الفواتير الشهرية (Bills.prg) الخاص بالاختيار (1) ، وبرنامج تحديد الحالة (ARStat.prg) الخاص بالاختيار (4) ، وهما يستخدمان في طباعة الفواتير (Invoices) . ولزيادة سرعة تشغيل البرنامج وكفائه يقوم كل من هذين البرنامجين بتشغيل ملف خطوات . وهذا الملف يتم تسميته (BillProc.prg) . وهو يحتوي على برنامج الخطوات (PrintBills.prg) بالإضافة إلى برنامج آخر يسمى (RowCheck) وهو يقوم باختبار رقم السطر فإذا زاد عدد السطور عن طول الصفحة يتم الانتقال إلى الصفحة التالية . وقد كان يمكن كتابة هذه البرامج كلها في ملف الخطوات (Proclib1) السابق إنشاؤه ولكن ذلك سوف يتطلب استخدام معالج كلمات آخر غير معالج الكلمات المستخدم مع (DBase III+) . وليست هناك حاجة لهذا في برنامج حسابات العملاء حيث يمكن فتح ملف الخطوات المطلوب وقت الحاجة ثم إغلاقه بعد ذلك . ولذلك فإن تعدد ملفات الخطوات لا يؤثر في كفاءة البرنامج .

والملف (BillProc.prg) يتكون من السطور التالية :

```
*****BillProc.prg

* Print a bill,using procedures PrintBills and
* Rowcheck Called from Bills.prg and ARstat.prg
PROCEDURE PrintBills
PARAMETERS M_Cust_No , Printer, Status

* - - - IF printer , print address, date in English
CLEAR
Page = 1
IF Printer
    EngDate = CMONTH (DATE()) + STR(DAY(DATE()), 3) + ;
        " , " + STR(YEAR(DATE()), 4)
    @ 1,0 SAY "My company , inc."
    @ 1,60 SAY EngDate
    ROW = 6
ELSE
    ROW = 1
ENDIF

* - - - -Print Customer name and address Using
* - - - -Row variable to control display and eject.
SELECT 1
IF Cus_Name # " "
    @ Row,0 SAY Cus_Name
ENDIF
@ Row+1,0 SAY Address
Row = Row + 4

* - - -Print customer number , terms and statrtting
* - - -balance from the customer file .
@ Row,0 SAY "Customer No, : " + STR(Cust_No, 4)
@ Row+1,0 SAY "Terms : " + Terms
@ Row+2,0 SAY "Balance of " + DTOC(LAST UPDAT) + ":"
@ Row+2,25 SAY Start_Bal PICT "999,999.99"
@ Row+3,0 SAY Uline
Start = Start_Bal
Row = Row + 5

* - - - - - Print heading for charges .
@ Row,0 SAY "Inv. # Part Description QTY"
@ Row,43 SAY "Price Total Date "
```

```

Row = Row + 2
* - - Select charges database :list and total current
* - - charges.
SELECT 2
SEEK M_Cust_No
Tot_Charge = 0
DO WHILE Cust_No = M_Cust_No .AND. .NOT. EOF()
    IF Status_.OR. .NOT. Billed
        @ ROW,0 SAY Invoice_No
        @ ROW,7 SAY Part_No
        @ ROW,13 SAY Descript
        @ ROW,34 SAY Qty
        @ ROW,39 SAY Unit_Pric PICT "999,999.99"
        @ ROW,50 SAY Amount PICT "999,999.99"
        @ ROW,62 SAY Date
        Tot_Charge = Tot_Charge + Amount
        ROW = ROW + 1
        * - - - - Make as billed if not a status check
        IF .NOT. STATUS
            REPLACE Billed WITH .T.
        ENDIF
    ENDIF(not status report and not already billed)
    SKIP
ENDDO (Cust_No = M_Cust_No)
* - - - - - Print payments heading .
@ ROW+1,0 SAY Uline
@ ROW+2,1 SAY "Payments / Adjustments"
ROW = ROW + 4

* - - - Check row position if displayed on screen.
DO RowCheck WITH 20,64
* - - - Select Payments file : list and total
* - - - payments/ adjustments.
SELECT 3
SEEK M_Cust_No
Tot_Pay = 0
DO WHILE Cust_No = M_Cust_No .AND. .NOT. EOF()
    IF Status_.OR. .NOT. Posted
        @ Row,0 SAY "Check #"
        @ Row,9 SAY Check_No
        @ Row,16 SAY Descript
        @ Row,50 SAY Amount PICT "999,999.99"
        @ Row,62 SAY Date
        Tot_Pay = Tot_Pay + Amount
    
```

```

        ROW = ROW + 1
        * - - Check row position if displayed on screen
        DO RowCheck WITH 20,64
        * - - Mark as billed if not status report
        IF .NOT. Status
            REPLACE Posted WITH .T.
        ENDIF
        ENDIF(not status report and not already posted)
        SKIP
    ENDDO(while Cust_No = M_Cust_No)

    * - - - Check row position if displayed on screen.
    @ Row,0 SAY Uline
    DO RowCheck WITH 17,56

    * - - Print starting balance ,total charges, payments,
    * - - ending balance , and thank you note.
    SET FIXED ON
    @ Row+1,5 SAY "Previous balance :"
    @ Row+1,25 SAY START PICT "999,999.99"
    @ Row+2,5 SAY "Total charges :"
    @ Row+2,25 SAY Tot_Charge PICT "999,999.99"
    @ Row+3,5 SAY "Payments received"
    @ Row+3,25 SAY Tot_Pay PICT "999,999.99"
    @ Row+4,5 SAY "Balance due      :"
    @ Row+4,25 SAY (Start + Tot_Charge) - Tot_Pay ;
    PICT "999,999.99"
    SET FIXED OFF
    IF Printer .AND. Status
        @ Row+8,10 SAY * * * Duplicate Invoice * * *
    ENDIF
    IF Pritner .AND. .NOT. Status
        @ Row+8,5 SAY "Thank you"
    ENDIF
    * - - - - - Pause if not going to the printer.
    IF .NOT. Printer
        @ 22,0 CLEAR
        WAIT
    ENDIF

    * - - - - - Done printing bill, Return to menu.
    RETURN

```



```
* - - - Procedure for checking row positions on screen
* - - - or printer
PROCEDURE RowCheck
PARAMETERS ScreenMax, PrintMax
IF .NOT. Printer .AND. Row >= ScreenMax
    @ 23,0 CLEAR
    WAIT "Press any key for next page"
    Row = 1
    CLEAR
ENDIF (Row too big for screen)
IF Printer .AND. Row >= PrintMax
    @ Row+2,70 SAY "Page" + STR(Page, 1)
    Page = Page + 1
    EJECT
    ROW = 5
ENDIF (Row too big for printer)
RETURN
```

ويلاحظ في البرنامج استخدام متغير الذاكرة (Status) لطباعة تقارير الحالة (Status) دون أن يؤثر ذلك على الفاتورة الشهرية . فإذا تم ادخال القيمة (T.O.) الى المتغير (Status) فان هذا يعني أن المطلوب تقرير حالة فقط وليس تقرير شهري، ولذلك تتم طباعة التقرير دون ترحيل . أما اذا كان التقرير فاتورة شهرية (Not Status) فيتم طباعته مع تحويل حقل الترحيل الى (False) حتى يتم ترحيله بعد ذلك باستخدام الاختيار (6) من القائمة الرئيسية لبرنامج حسابات العملاء .

ويجدر العلم أن الملف السابق ما هو الا ملف خطوات يتم استدعاؤه بواسطة برنامج التقارير الشهرية (Bills.prg) وبرنامج تحديد الحالة (ARStat.prg) . وهذان البرنامجان سيتم شرحهما في الأجزاء التالية .

٢١ - ٢ برنامج الفواتير الشهرية (Bills.prg)

يستخدم هذا البرنامج عند ادخال الاختيار رقم (1) في قائمة برنامج طباعة تقارير حسابات العملاء . وهذا البرنامج يقوم بطباعة الفواتير الشهرية وفي نفس الوقت يقوم بادخال القيمة (F.O.) في حقل الترحيل سواء كان حقل التسديد (Billed) في ملف الصرف (Charges.dbf) ، أو حقل التسديد (Posted) في ملف السداد (Payments.dbf) . وذلك حتى يتسنى للمستخدم بعد ذلك

ترحيل هذه الفسوايتر الى الملف الرئيسي عن طريق برنامج الترحيل الشهري (Monthly Posting) بالاختيار رقم (٦) من القائمة الرئيسية لبرنامج حسابات العملاء .

ويتكون هذا البرنامج من السطور التالية :

```

.....Bills.prg

*      Prints monthly statemetns.
*      Called from AR Print menu.

* - - - - Have user prepare printer (or cancel).
Proceed = " "
DO Title WITH "Print Monthly Bills "
@ 15,5 SAY "Repare printer and press a key to proceed"
@ 17,5 SAY "(Type X to cancel)" GET Proceed PICT "|"
READ
* - - - - Return to menu if requested.
IF Proceed = "X"
    RETURN
ENDIF

* - - - Open files and delete records with "0" Amount.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Charges INDEX ChargNo
DELETE ALL FOR Amount = 0
SELECT 3
USE Payments INDEX PayNo
DELETE ALL FOR Amount = 0 ,

* - - - Set decimal place to 2 , send @ ... SAY to
* - - - printer and open bill_printing procedure file.
CLEAR
SET DECIMALS TO 2
SET DEVICE TO PRINT
SET PROCEDURE TO BillProc

* - Set parameters to printer and "not status" report.
```

```

Printer = .T.
Status = .F.

* - - - Loop through Customer database and print a
* - - - bill for every one
SELECT 1
DO WHILE .NOT. EOF()
    Lookup = Cust_No
    DO PrintBills WITH Lookup , printer , status
    EJECT

    * - - - - - Set next customer
    SELECT 1
    SKIP
ENDDO

* - - - - - Done , Close files
SET DEVICE TO SCREEN
CLOSE DATABASES
CLOSE PROCEDURE
* - - - - - Open ProcLib1 procedure file
SET PROCEDURE TO ProcLib1
* - - - Print reminder about posting , then return to
* - - - main menu .
CLEAR
TEXT
    monthly postings (main menu option 5) should be
    performed immediately after printing the monthly
    statements.
ENDTEXT
?

@ 22,10 SAY "Press any key to return to main menu..."
WAIT " "
RETURN TO Master
    
```

ويلاحظ في هذا البرنامج استخدام ملف الخطوات (BillProc.prg) لطباعة التقارير الشهرية . كما يلاحظ فتح ملف الخطوات الآخر (ProcLib1) قبل نهاية البرنامج حتي يستخدم في باقي البرامج التي سوف تحتاجه ؛ كما يلاحظ استخدام الأمر (RETURN TO MASTER) للرجوع الى القائمة الرئيسية لبرنامج حسابات العملاء مباشرة حتى يتسنى للمستخدم استخدام الاختيار رقم (6) في ترحيل الفواتير .

٢١ - ٤ برنامج اختبار الحالة (ARStat.prg)

يستخدم هذا البرنامج عند ادخال الاختيار رقم (٤) في قائمة برنامج طباعة تقارير حسابات العملاء ، وهو يسمح للمستخدم باختبار حالة العميل من حيث تسديد الفواتير (Billed) او عدم تسديدها ، وهو يتيح له الحصول على صورة أخرى من الفاتورة المرسلة الى العميل لتسديدها . ويلاحظ في هذا البرنامج اعطاء المتغير (Status) القيمة (True) حتى يعرف برنامج الخطوات (PrintBills) أن المطلوب هو اختبار الحالة فقط وليس الفاتورة الحقيقية . وهذا البرنامج يسمى (ARStat.prg) ويتكون من السطور التالية :

```

.....*ARStat.prg
* Quick lookup of a single statement.
* Called from A/R Print menu.

* - - Open files and delete records with "0" amounts.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Charges INDEX ChrgNo
DELETE ALL FOR Amount = 0
SELECT 3
USE Payments INDEX PayNo
DELETE ALL FOR Amount = 0

* - - - Set up memory variables for status report.
Status = .T.
Printer = .F.
M_Cust_No = 0
M_Name = " "
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - - - Print screen title
    DO Title WITH "Quick lookup of current status"
    * - - - - Get Customer by number or name
    SELECT 1
    DO GetCust WITH ;
        M_Cust_No , M_Name, M_Address, Exiting
    * - - - - Proceed with bill.
    IF .NOT. Exiting
        * - - - - Ask about printer.
        @ 5,0 CLEAR
        LP = " "
```

```
@ 15,5 SAY "Send statement to printer ? Y/N ;
GET LP PICT "I"
READ
CLEAR
* - - - - - Set up printer if necessary
IF LP = "Y"
    Printer = .T.
    SET DEVICE TO PRINT
ENDIF

* - - - - - Print current statement for customer.
SET PROCEDURE TO BillProc
DO PrintBill WITH M_Cust_No , Printer, STATUS
CLOSE PROCEDURE
SET PROCEDURE TO ProcLib1
* - - - - - turn off printer
IF Printer
    EJECT
    SET DEVICE TO SCREEN
ENDIF
ENDIF (not exiting)
* - - - - - Close files and return to main menu
CLOSE DATABASES
RETURN
```

٢١ - ٥ برنامج التقارير المختصرة والزمنية (AgeSumm.prg)

الاختياران (2) ، (3) من قائمة برنامج طباعة تقارير حسابات العملاء يسمحان للمستخدم بالحصول على تقارير شهرية مختصرة وتقارير زمنية (Aging Reports) . وهذه التقارير يتم انشاؤها عادة بعد طباعة الفواتير الشهرية مباشرة .

والتقارير الشهرية المختصرة (Summary Reports) تعرض الموازنة الحالية للمييل وموقف الصرف والسداد الخاص به . حيث يتم عرض تقرير بالصورة الموضحة بالشكل (٢١ - ٢) .

ويتم انشاء صورة التقرير باستخدام الأمر (CREATE REPORT) وذلك كالآتي :

```
USE Customer
CREATE REPORT ARSumm
```

Page No. 1

11/21/90

Monthly Activity Summary

Cust. No.	Name	Current Balance	Current Charges	Current Payments	Last Update
001	Ahmed Salem	80.00	180.00	100.00	01/30/90
021	Tarak Fahy	0.00	0.00	150.00	01/10/90
■ ■ Total ■ ■		80.00	180.00	250.00	

شكل (٢ - ٢١)

وفي هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التي يتم من خلالها تحديد محتويات الأعمدة (Columns). وذلك كما هو موضح بالشكل (٢ - ٢١).

	Contents	Heading	Width	Decimal	Total
1	Cust. No.	Cust. No.	5	0	N
2	Time(Cust. Name)	Name	26	0	
3	Start Bal	Current : Balance	7	2	Y
4	Chg. Earn	Current : Charges	8	2	Y
5	Pay. Curr	Current : Payments	8	2	Y
6	Last Update	Last. Posted	8	0	

شكل (٢ - ٢١)

أما التقارير الزمنية فتسمح للمستخدم بعرض الموازنة الخاصة بكل عميل في فترات مختلفة لمراقبة موقف هذا العميل ، حيث يكون التقرير بالصورة الموضحة بالشكل (٢١ - ٤) .

Page No. 1 02/30/90 <i>Accounts Receivable Aging Report</i>					
Name	Current	30 day	60 day	90 day	90+ day
Customer Number: 1000 Ahmed Saleh	80.00	88.00	95.00	0.00	0.00
Customer Number: 1021 Alik Fahy	0.00	98	90	120	200

شكل (٢١ - ٤)

ولإنشاء هذا التقرير يتم استخدام الأمر (CREATE REPORT) كالاتي :

USE Customer
CREATE REPORT Aging

وفي هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التي يتم من خلالها تحديد محتويات الأعمدة (Columns) وذلك كما هو موضح بالشكل (٢١ - ٥) .

ولعرض رقم العميل مع اسمه يستخدم الاختيار (Group) ويتم استخدام رقم العميل حقلاً للتجميع بناءً عليه وكتابة عنوان المجموعة (Customer Number) وبذلك يظهر هذا العنوان يليه رقم العميل مع كل عميل كما يتضح من صورة التقرير السابق ايضاحها .

	Contents	Heading	Width	Decimal	Total
1	Trn(Cash_Name)	Name	20		
2	Start_Bal	Current Balance	9	2	N
3	Bal_30	30 days	8	2	N
4	Bal_60	60 days	8	2	N
5	Bal_90	90 days	8	2	N
6	Bal_90Plus	90 + days	8	2	N

شكل (٢١ - ٥)

ويتم طباعة التقارير المختصرة (Summary Reports) والتقارير الزمنية (Aging Reports) باستخدام البرنامج (AgeSum.prg) مع استخدام الماكرو لتحديد نوع التقرير المطلوب . ويتضح ذلك من الأوامر التالية الموجودة في برنامج الطباعة (ARPrint.prg) .

```

CASE PChoice = 2
    RepForm = "ARSumm"
    DO AgeSumm
CASE PChoice = 3
    RepForm = "Aging"
    DO AgeSumm
    
```

بعد اختيار المستخدم للرقم (2) يتم تخزين كلمة (ARSumm) في متغير الذاكرة (RepForm) . وعند اختياره للرقم (3) يتم تخزين كلمة (Aging) في نفس هذا المتغير . وعند تنفيذ البرنامج (AgeSum) في الحالتين يتم طباعة التقرير الخاص بكل حالة .

ويتم كتابة سطور هذا البرنامج كالآتي :


```

* * * * * AgeSumm.prg
*   Print aged balances or summary reports.
*   Called from A/R Print menu.
@ 5,0 CLEAR

* - - - - Ask about printer
STORE " " TO LP, Mac
@ 15,5 SAY "Send report to printer ? (Y/N)" ;
  GET LP PICT "!"
READ
* - - - - Set up the printer if necessary.
IF LP = "Y"
  Mac = "TO PRINT"
  WAIT "Prepare printer, then press any key to print"
ENDIF

* - - - - Use Customer database and report stored in
* - - - - RepForm
CLEAR
USE Customer INDEX CustNo
REPORT FORM & RepForm & Mac

* - - - - Pause , if necessary , then return to menu
IF LP # "Y"
  WAIT "Press any key to return to AR Print menu."
ELSE
  EJECT
ENDIF
RETURN

```

وعند تنفيذ هذا البرنامج يتم التعويض بنوع التقرير المطلوب سواء كان تقريراً مختصراً (ARSumm) أو زمنياً (AgeSumm) مكان المتغير (RepForm) حسب اختيار المستخدم . كما يتم التعويض بالسلسلة الحرفية (TO PRINT) مكان المتغير (Mac) إذا اختار المستخدم الطباعة . وهذا يؤدي الى أن يصبح الأمر في الصورة التالية :

REPORT FORM ARSumm TO PRINT

وذلك في حالة اختيار المستخدم للاختيار رقم (2) من قائمة الطباعة . وبالتالي

يتم طباعة التقرير المطلوب .

٢١ - ٦ التقارير التاريخية (ARHist.prg)

التقارير التاريخية تسمح للمستخدم بالبحث عن أى بيانات سابقة للعميل خلال أى فترة زمنية . ويتم الدخول فى قائمة التقارير التاريخية عندما يختار المستخدم الاختيار رقم (5) من القائمة الرئيسية لبرنامج طباعة تقارير حسابات العملاء . وفى هذه الحالة تظهر القائمة الموضحة بالشكل (٢١ - ٦) .

<i><u>History Menu</u></i>	<i><u>02/20/90</u></i>
<ol style="list-style-type: none"> 1. Search by Customer Code 2. Search by Product Code 3. Search by Date 4. Return to Main Menu 	
<p>Enter Choice (1 - 4)</p>	

شكل (٢١ - ٦)

وعندما يختار المستخدم الاختيار رقم (1) من القائمة ثم يقوم بكتابة رقم العميل المطلوب ، تظهر جميع البيانات التاريخية الخاصة بهذا العميل بالصورة الموضحة بالشكل (٢١ - ٧) . ويلاحظ من هذا الشكل أن التقرير مكون من جزئين ، جزء علوى وجزء سفلى . وهذا يتطلب انشاء صورتين للتقرير باستخدام الأمر (CREATE REPORT) . حيث يتم طباعة الجزء العلوى من التقرير باستخدام الملف (CusHist1.frm) ويتم انشاؤه عن طريق كتابة السطرين التاليين :

```
USE ChrgHist
CREATE REPORT CusHist1
```

ويلاحظ هنا استخدام الملف (ChrgHist.dbf) الذي سبق انشاؤه قبل كتابة برنامج حسابات العملاء .

وفي هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التي يتم من طريقها تحديد عنوان التقرير (Customer History) . كما يتم تحديد محتويات الأعمدة كالآتي :

(Check_No) ، (Part_No) ، (Qty) ، (Descript) ، (Unit_Price) ، (Amount) ، (Date) .

كما يتم تحديد عناوين الأعمدة كما هو واضح من الشكل (٢١ - ٧) .

Page No. 1 02/20/90						
Customer History						
Part No.	Part No.	Qty	Description	Unit Price	Total	Date
1000	BBB	5	Floppy disks	1500	80	01/30/90
1000	AAA	2	Printer	980	1000	02/05/90
Check No.	Check No.	Description	Amount	Date		
1000	1333	Payment	100.00	02/05/90		
1000	1750	Payment	288.00	02/30/90		

شكل (٢١ - ٧)

أما الجزء السفلي من التقرير فهو يتعلق بموقف التمسديد الخاص بهذا العميل . ويتم تكوينه بواسطة الملف (CusHist2.frm) الذي يتم انشاؤه بكتابة السطرين التاليين .

USE PayHist
CREATE REPORT CusHist2

وفي هذه الحالة تظهر قوائم برنامج المساعد كما سبق الايضاح ويتم من خلالها تحديد محتويات الأعمدة كالآتي :

(Cust_No) ، (Check_No) ، (Descript) ، (Amount) ، (Date)

كما يتم تحديد عناوين هذه الأعمدة كما هو واضح من الشكل (٢١ - ٧) .

وإذا اختار المستخدم الاختيار رقم (2) من قائمة التقارير التاريخية فإن هذا يعني أن التقرير المطلوب من صنف معين . ولذلك يتم ادخال رقم هذا الصنف (Part_no) . وفي هذه الحالة يظهر التقرير الموضح بالشكل (٢١ - ٨) .

Product Code History					
Part No	Description	Qty	Unit Price	Total	Date
AAA	Printer	2	900	1800	02/05/90
AAA	Printer	3	900	2700	02/10/90
AAA	Printer	5	900	4500	02/20/90
Total		10	900	8200	

شكل (٢١ - ٨)

ويتم انشاء هذا التقرير من طريق كتابة السطرين التاليين :

```
USE BillHist
CREATE REPORT CodeHist
```

وعند ظهور قوائم الاختيارات الخاصة ببرنامج المساعد (Assistant) يتم كتابة عنوان التقرير (Heading) كالآتي :

(Product Code History)

كما يتم تحديد محتويات الأعمدة (Columns) كالآتي :

(Part_No), (Descript), (Qty), (Unit_Price), (Amount), (Date)

ويفيد هذا التقرير في متابعة موقف كل صنف ومعرفة معدل صرفه .

وإذا اختار المستخدم الرقم (3) من قائمة التقارير التاريخية فإن هذا يعني أن المستخدم يريد طباعة تقرير عن موقف الموازنة في تاريخ محدد . وفي هذه الحالة يقوم بإدخال التاريخ المطلوب .

وهذا التقرير يتكون من جزأين . الجزء العلوي يتم انشاؤه بواسطة الملف (DatHist1.frm) ويتم ذلك عن طريق كتابة السطرين التاليين :

```
USE BillHist
CREATE REPORT DatHist
```

ويتم كتابة العنوان (Product Code History) للجزء العلوي ، كما يتم تحديد محتويات الأعمدة (Columns) كما يلي :

(Date), (Part_No), (Descript), (Qty), (Unit_Price), (Amount) ,
(Cust_No)

والجزء السفلي يتم انشاؤه بواسطة الملف (DatHist2.frm) . ويتم ذلك بكتابة السطرين التاليين :

```
USE PayHist
CREATE REPORT DatHist2
```

ويتم تحديد محتويات الأعمدة كالآتي :

(Date) , (Check_No) , (Amount) , (Descript) , (Cust_No) .

ويجب ملاحظة أن هذه المجموعة من التقارير هي مجرد اقتراحات . ويستطيع مخطط البرامج انشاء أى تقارير أخرى حسب الحاجة . وبعد انشاء التقارير المختلفة يتم كتابة البرنامج الذى يؤدى الى طباعة كل نوع من هذه التقارير . وذلك كالآتى :

```

***** ARHist.prg,
* Search history, and current charges and payments
* files and display summary data.
* Called from AR Print Options menu.
HChoice = 0
DO WHILE HChoice # 4
  CLEAR
  DO Title WITH "History Menu"
  TEXT
    1. Search by customer code
    2. Search by product code
    3. Search by date
    4. Return to main menu.
  ENDTEXT
  @ 24,1 SAY "Enter choice (1 - 4) "
  GET HChoice PICT "9" RANGE 1,4
  READ
  * - - - - - Set up search macro accordingly.
  @ 4,0 CLEAR
  DO CASE
    CASE HChoice = 1
      M_Cust_No = 0 ;
      @ 15,5 SAY "Enter customer number"
      GET M_Cust_No PICT "99999"
      READ
      LookAT = "Cust_No"
      LookFOR = M_Cust_No
      RepForm1 = "CusHist1"
      RepForm2 = "CusHist2"
      SET EXACT ON
    CASE HChoice = 2

```

```

M_Code = SPACE (5)
@ 15,5 SAY "Enter product code" GET M_Code
READ
LookAT = "UPPER(Part_No)"
LookFOR = UPPER(M_Code)
RepForm1 = "CodeHist"
SET EXACT ON
CASE Hchoice = 3
M_Dat = SPACE (8)
@ 15,5 SAY "Enter Date " GET M_Date
READ
LookAT = "DTC(Date)"
LookFOR = TRIM(M_Date)
RepFlrm1 = "DatHist1 "
RepForm2 = "DatHist2"
SET EXACT OFF
CASE Hchoice = 4
SET EXACT OFF
RETURN
ENDCASE
* - - - - - Ask about printer.
@ 5,0 CLEAR
STORE " " TO Lp, Mac
@ 15,5 SAY "Send report to printer ? (Y/N)" ;
GET LP PICT "!"
READ
IF Lp = "Y"
Mac = "TO PRINT"
ENDIF
* - - - - - Search billing History file
USE BillHist
SET FILTER TO &LookAt = LookFor
COPY TO Temp
USE Temp
APPEND FROM Charges FOR & LookAT = LookFOR
SET FILTER TO &LookAT = LookFOR
REPORT FORM &RepForm1 &Mac

```

```
* - - - if not searching for product code , Search
* - - - Payments files
IF HChoice # 2
    USE PayHist
    SET FILTER TO &LookAT = LookFOR
    REPORT FORM &RepForm2 &Mac PLAIN NOEJECT
ENDIF
* - - - IF report not going to printer, pause.
IF Lp # "Y
    ?
    ?
    WAIT
ENDIF
ENDDO(HChoice # 4)
```


الفصل الثاني والعشرون

التحديث الشهرى للنظام

يتم تحديث النظام مرة واحدة كل شهر عندما يختار المستخدم الرقم (6) من القائمة الرئيسية لبرنامج حسابات العملاء حيث يتم تشغيل برنامج الترحيل الشهرى للنظام (Post.prg) الذى يقوم بالآتى :

١ - نقل بيانات الموازنة خلال ثلاثين يوما ، أو ستين يوما أو ٩٠ يوما أو أكثر من ذلك فى بيان الفترة السابقة على الترتيب وذلك باستخدام الأمر (REPLACE) بحيث تنتقل بيانات حقل الموازنة خلال (٢٠) يوما الى حقل الموازنة خلال (٦٠) يوما وبيان حقل الموازنة خلال (٦٠) يوما الى حقل الموازنة خلال (٩٠) يوما . وتضاف بيانات حقل الموازنة خلال (٩٠) يوما الى بيانات حقل الموازنة خلال أكثر من ٩٠ يوما .

٢ - جميع فواتير حركة الصرف التى تمت كتابتها يتم تلخيصها وإدخالها فى ملف قاعدة بيانات جديد يسمى (summary.dbf) . ثم يتم تحديث الملف الرئيسى (Master.dbf) مسنن الملف (summary.dbf) باستخدام الأمر (UPDATE FROM) .

٣ - كل السجلات الخاصة بالفواتير التى تم تسديدها (Billed = .T.) تضاف الى ملف الصرف التاريخى (BillHist.dbf) وتسمح من ملف حركة الصرف (charges.dbf) وذلك لتجهيز ملف حركة الصرف للشهر الجديد .

٤ - يتم تنفيذ نفس هذه العملية على ملف حركة التسديد (Payments.dbf) . حيث يتم نقل جميع السجلات التى تم ترجيلها (Posted = .T.) الى ملف التسديد التاريخى (PayHist.dbf) . كما يتم مسحها من ملف حركة التسديد (Payments.dbf) .

٥ - بعد الانتهاء من ترجيل سجلات الفواتير التى تم تسديدها يتم تعديل محتويات حقل الموازنة الابتدائية للشهر (Starting Balance) وكذلك حقل تاريخ آخر تحديث (Last Updating) باستخدام الأمر (REPLACE) .

٦ - يقوم البرنامج أيضا باستخدام كلمة مرور (Password) لتأمين عملية ترحيل البيانات حتى لا تتم بواسطة شخص غير مسئول . وقد تم استخدام كلمة (Mohamed) ككلمة مرور ولكن يمكن استخدام أى كلمة أخرى . ونظرا لأن عملية الترحيل الشهرى لحركة الصرف والتسديد قد تأخذ وقتا طويلا إذا كان الملف كبيرا جدا ، لذلك يتم كتابة السطور التالية :

```
*      - - - -Show progress
@ 20,1 SAY "Aging the balances : Record" + ;
      STR(RECNO(),4) + "OF" + STR(RECCOUNT(), 4)
```

وهذه السطور عند تنفيذها في البرنامج تؤدي الى ظهور الآتي على الشاشة :

Aging the balance : Record 2 of 100

ومع نقل كل سجل يزيد العدد التالي لكلمة (Record) بواحد حتى انتهاء عملية النقل . وهذه الرسالة تؤدي الى اطمئنان المستخدم ان عملية النقل تتم بدون مشاكل .

والبرنامج (Post.prg) يتم كتابته كالآتي :

```
*****Post.prg
*   Posts summarized monthly accounts to the Customer
*   file. Called from AR main menu .
SET DELETED ON
CLEAR
DO Title WITH "Monthly Posting"
* - - - - - Display and get password
TEXT
```

This is the program to post Payments and charges.
Be sure you have printed all the monthly invoices
before proceeding with this program.

```
ENDTEXT
Password = SPACE(7)
@ 15,12 SAY "Enter password to proceed" GET Password ;
      PICT "!!!!!!!"
```

READ

```
* - - - - If proper password not entered return to the
* - - - - - menu.
```

```
IF Password # "MOHAMED"
    ? "Illegal password" , CHR(7)
    SET DELETED OFF
    RETURN
ENDIF
```

```

* - - Do the posting,first,shift all current 30 , 60
* - - and 90 day billings "back" one field in the
* - - customer file.
@ 20,1 SAY "Working ...."
CLOSE DATABASES
USE Customer;
REPLACE ALL;
    Bal_90Plus WITH Bal_90Plus + Bal_90, ;
    Bal_90 WITH BAL_60, ;
    BAL_60 WITH Bal_30, ;
    Bal_30 WIHT Chg_Curr - Pay_Curr
REPLACE ALL; ,
    Chg_Curr WITH 0 , ;
    Pay_Curr WITH 0

* - - - Now,create summary of the charges database by
* - - - customer number.
SET SAFETY OFF
USE Charges INDEX ChrgNo
COPY STRUCTURE TO Summary
TOTAL ON Cust_No TO Summary FIELDS Qty , ;
    Unit_Price ,Amount FOR Billed
* - -Now update the customer database current balances
* - -with data from the charges summary file.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Summary
SELECT 1
UPDATE ON Cust_No FROM Summary REPLACE Chg_Curr WITH ;
    B -> Amount

* - - - Move all posted transactions to the billing
* - - history file
SELECT 2
USE BillHist
APPEND FROM Charges FOR Billed

```

```
* - - - - - Then empty the current charges file.
CLOSE DATABASE
USE Charges INDEX ChrgNo
DELETE ALL FOR Billed
PACK

* - - Now , summarize payment totals for each Customer
USE Payments INDEX PayNo
COPY STRUCTURE TO Summary
TOTAL ON Cust_No TO Summary Fields Amount FOR Posted

* - -Now update the customer database current balances
* - -with data from the payments summary file.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Summary
SELECT 1
UPDATE ON Cust_No FROM Summary REPLACE Pay_Curr ;
      WITH B -> Amount

* - - Append all posted transactions to the payments
* - - history file.
SELECT 2
USE PayHist
APPEND FROM Payments FOR Posted

* - - - - - Then empty the current payments file.
CLOSE DATABASES
USE Payments INDEX PayNo
DELETE ALL FOR Posted
PACK

* - - - Then update the 'last billed' and 'starting
* - - balance' Fields in the customer database.
USE Customer
REPLACE ALL Start_Bal WITH Start_Bal + Chg_Curr ;
      - Pay_Curr
```

```

REPLACE ALL Last_Updat WITH DATE()

* - - - - - Adjust aged balances.
GO TOP
DO WHILE .NOT. EOF()
    * - - - - - Show progress.
    @ 20,1 SAY "Aging the balances : Record" + ;
        STR(RECNO(), 4) + "OF" + STR(RECCOUNT(), 4)
    * - - - - - IF no payment, skip calculations.
    IF Pay_Curr <= 0
        SKIP
        LOOP
    ENDIF (Pay_Curr <= 0)
    * - - - - - Otherwise , subtract the payment
    More = .F.
    NextBal = .T.
    IF Bal_90Plus > 0
        Remain = Pay_Curr - Bal_90Plus
        IF Remain >= 0
            REPLACE Bal_90Plus WITH 0
            More = .T.
        ELSE
            REPLACE Bal_90Plus WITH ABS(Remain)
            NextBal = .F.
        ENDIF(Remain >= 0)
    ENDIF(90+ > 0)
    * - - - - - 90 days.
    IF NextBal .AND. Bal_90 > 0
        IF More
            Remain = Remain - Bal_90
        ELSE
            Remain = Pay_Curr - Bal_90
        ENDIF(More)
        IF Remain >= 0
            REPLACE Bal_90 WITH 0
            More = .T.
        ELSE
            REPLACE Bal_90 WITH ABS(Remain)

```

```

        NextBal = .F.
    ENDIF(Remain >= 0)
ENDIF (NextBal & Bal90 > 0)
* - - - - - 60 days.
IF NextBal .AND. Bal_60 > 0
    IF More
        Remain = Remain - Bal_60
    ELSE
        Remain = Pay_Curr - Bal_60
    ENDIF(More)
    IF Remain > = 0
        REPLACE Bal_60 WITH 0
        More = .T.
    ELSE
        REPLACE Bal_60 WITH ABS(Remain)
        NextBal = .F.
    ENDIF(Remain >= 0)
ENDIF (NextBal & Bal60 > 0)
* - - - - - 30 days.
IF NextBal .AND. Bal_30 > 0
    IF More
        Remain = Remain - Bal_30
    ELSE
        Remain = Pay_Curr - Bal_30
    ENDIF(More)
    IF Remain > = 0
        REPLACE Bal_30 WITH 0
        More = .T.
    ELSE
        REPLACE Bal_30 WITH ABS(Remain)
        NextBal = .F.
    ENDIF(Remain >= 0)
ENDIF (NextBal & Bal30 > 0)
SKIP
ENDDO (end of file)

```


* - - - - - Display closing messages

CLEAR

? CHR(7)

TEXT

The posting procedure is complete. Use option 4, from the main menu to print current monthly summary and aging reports.

ENDTEXT

* - - - - - Get rid of any old keypresses.

CLEAR TYPEAHEAD

WAIT "Press any key to return to main menu"

* - - - - - Return to the main menu.

SET DELETED OFF

CLOSE DATABASES

RETURN

ويلاحظ في البرنامج استخدام عدة أوامر (IF) في تنفيذ الحسابات الخاصة بكل حالة . فمثلا إذا لم يتم تسديد أى قيمة جديدة أى (Pay Curr <= 0) ، فان هذا يعنى أنه ليست هناك حسابات مطلوب إجراؤها . ولذلك يتم الانتقال الى العميل التالى . وخلاف ذلك يتم ضبط حسابات (٢٠) يوما ، (٦٠) يوما ، (٩٠) يوما ، وأكثر من ٩٠ يوما لكل عميل .

ويتم استخدام المتغير (Remain) فى عملية الضبط . فاذا كان المبلغ الذى تم تسديده يغطي الحسابات القديمة أكثر من ٩٠ يوما ، فى هذه الحالة يتم تصفير هذه الحسابات (جعلها تساوى صفرا) . وهكذا يتم المرور على باقى الحسابات (٩٠) يوما ، (٦٠) يوما ، (٢٠) يوما على الترتيب .

وعند الانتهاء من حساب جميع العملاء يتم عرض رسالة للمستخدم عن التقارير المطلوبة . ثم يتم اغلاق جميع الملفات والعودة الى القائمة الرئيسية .

الباب الثالث والعشرون

برنامج التكامل بين حسابات العملاء والمخازن

يستخدم هذا البرنامج في ربط برنامج حسابات العملاء ببرنامج المخازن. ولتنفيذ ذلك يجب اجراء التعديلات التالية :

- ١ - يتم طرح كميات الأصناف التي تم ادخالها في حركة الصرف من كميات الأصناف الموجودة في ملف المخازن الرئيسي (Master.dbf).
- ٢ - يتم تمييز السجلات التي يتم ترحيلها الى ملف المخازن الرئيسي (Master.dbf) بعلامة معينة حتى لا يتم ترحيلها مرة أخرى.
- ٣ - يتم اختبار حركة الصرف (Charges) في برنامج حسابات العملاء أو حركة السداد (Payemnts) قبل ترحيلها الى الملف الرئيسي للمخازن .

وفي البداية يجب اضافة حقل جديد الى ملف حركة الصرف (Charges.dbf) يوضح للبرنامج اذا كان السجل قد تم ترحيله الى ملف المخازن أم لا . وهذا الحقل يكون حقلا منطقيا ونسميه (InPost) حيث يمثل الحرفان (In) أول حرفين في كلمة (Inventory) ويتم ذلك باستخدام الأمر (MODIFY STRUCTURE). كما يتم اضافة نفس الحقل الى ملف الصرف التاريخي (BillHist.dbf) حتى يصبح بنفس التركيب . حيث أن هذا الملف يستقبل السجلات التي يتم ترحيلها الى ملف حسابات العملاء الرئيسي (Customer.dbf).

ولتأمين عملية الترحيل يجب التأكد أن كل سجل جديد يتم اضافته الى ملف حركة الصرف (Charges.dbf) يتم تمييزه حتى يعلم البرنامج أنه لم يتم ترحيله الى ملف المخازن الرئيسي (Master.dbf). ويتم ذلك عن طريق اضافة سطر معين الى برنامج اضافة العملاء (NewChrg.prg) في الجزء الخاص باضافة السجل الجديد . حيث يتم اضافة السطر التالي :

```
REPLACE InPost WITH .F.
```

ويصبح هذا الجزء من البرنامج كالآتي :

```
IF .NOT. Exiting
  SELECT 2
  APPEND BLANK
  REPLACE Cust_No WITH M_Cust_No
  REPLACE Date WITH DATE()
  REPLACE Billed WITH .F.
  REPLACE InPost WITH .F.
  SET FORMAT TO FNewChrg
  READ
  CLOSE FORMAT
  REPLACE Amount WITH Qty * Unit_Price
ENDIF
```

ثم يتم كتابة البرنامج الذي يقوم بتحديث ملف المخازن الرئيسي من كل السجلات التي لم يتم ترحيلها (InPost=.F.) من ملف حركة الصرف (Charges.dbf) وملف الصرف التاريخي (BilHist.dbf) كما يقوم بتغيير حقول التحديث (Inpost) الى (True) حتى لا يتم ترحيله مرة ثانية .

وحيث أن البرنامج (Updater.prg) هو البرنامج الذي يقوم بتحديث ملف المخازن الرئيسي . لذلك يتم اضافة سطر الى هذا البرنامج يؤدي الى تشغيل برنامج آخر اسمه (ARUpdate.prg) يختص بتحديث الملف الرئيسي من ملف حركة الصرف . ويتم اضافة هذا السطر قبل نهاية ملف التحديث (Updater) كالآتي :
مثلا :

DO ARUpdate

والبرنامج (ARUpdate.prg) يتكون من السطور التالية :

```
*****ARUpdate.prg
*   Update the master file from charges and BilHist.
CLEAR
? "Updating from the A/R System .."
* - - - - - Use the charges databases.
USE Charges
* - - - - - Copy nonupdated records to Temp File.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. InPost
* - - - - - Now get the recrods from BillHist.
USE Temp
APPEND FROM BillHist FOR .NOT. InPost
* - - - - - Get the Temp file sorted by part number.
INDEX ON Part_No TO ARIndex

* - - - - - Use the Master file for updating .
SELECT 1
USE Master INDEX Master
SELECT 2
USE Temp INDEX ARIndex
```

```
* - - - Update Master from the temporary A/R file.
SELECT 1
UPDATE ON Part_No FROM Temp REPLACE Qty WITH ;
      Qty - Temp -> Qty

* - - - Use the origingal charges database file
* - - - change all posted fields to true
CLOSE DATABASES
USE Charges
REPLACE ALL InPost WITH .T.

* - - - - - Do the same thing with BillHist file.
USE BillHist
REPLACE ALL InPost WITH .T.
* - - - - - Return to the updater command file.
RETURN
```


5

الجزء الخامس

بعض الأدوات المتقدمة

- * برنامج كتابة الشيكات
- * برنامج اختيار الالوان
- * برنامج تحريك العمود الضوئى

مقدمة

هذا الجزء يقدم مجموعة من الأدوات المتقدمة (Advanced Tools) التي يمكن لمخطط البرامج استخدامها في كثير من التطبيقات . وهذه الأدوات بالإضافة إلى ما توفره للمستخدم من جهد ووقت ، فإنها أيضا تزيد من كفاءة البرنامج الذي يجري أعداده .

و يتكون هذا الجزء من ثلاثة فصول . الفصل الأول يشرح برنامج يمكن المستخدم من طباعة الشيكات مع القدرة على كتابة مبلغ الشيك بالأرقام و الحروف . و الفصل الثاني يتيح لمخطط البرامج استخدام الألوان في الشاشات و القوائم التي يتم عرضها على المستخدم . كما يتيح للمستخدم اختيار الألوان المناسبة له من خلال قائمة اختيارات خاصة . و الفصل الثالث يتيح لمخطط البرامج تصميم قوائم اختيارات تسمح للمستخدم بتحريك مؤشر على الشاشة إلى الاختيار المطلوب ، و ذلك علاوة على الطريقة التقليدية في الاختيار عن طريق الأرقام أو الحروف .

الفصل الرابع و العشرون

برنامج كتابة الشيكات

في معظم الأحيان يحتاج مخطط البرامج الى طباعة الشيكات من خلال البرنامج .
وعادة ما يحتوى الشيك على الرقم المثل للمبلغ المطلوب صرفه ، كما يحتوى أيضا
على ترجمة انجليزية لهذا الرقم . فمثلا الرقم (950) يتم كتابته كالآتي :

(Nine Hundred and Fifty)

ولكن الأعداد التي يتم الحصول عليها من برنامج قاعدة البيانات تكون دائما على
الصورة العددية .

وحيث أن البرنامج لايعرف مقدما القيمة العددية التي يتم كتابتها في الشيك .
لذلك يصبح في منتهى الصعوبة على مخطط البرامج تحويل كل عدد الى الصورة
الحرفية عند كتابته في الشيك . ولذلك فإن هذا البرنامج يتيح لمخطط البرامج ترجمة
كل عدد من (1) حتى (999,999.99) الى الأعداد الحرفية المقابلة .

ولتوضيح ذلك يتم أولا انشاء ملف قاعدة بيانات لكتابة الشيك حتى يتم من
طريقه اختبار البرنامج بعد ذلك . ويتكون هذا الملف من الحقول التالية :

Field	Field Name	Type	Width	Dec
1	CHECK_NO	Numeric	5	0
2	TO_WHOM	Character	25	
3	AMOUNT	Numeric	9	2
4	DATE	Date	8	
** TOTAL **			48	

شكل (٢٤ - ١)

وقبل كتابة البرنامج يجب أولا تخزين المقابل الحرفي لكل عدد من الأعداد من واحد إلى عشرين في متغيرات ذاكرة وكذلك أرقام العشرات مثل (Thirty) ، (Fourty) ، الخ . كما يتم تخزين هذه المتغيرات في ملف ذاكرة . (Memory File) وهذا الملف نسميه (English.mem) ويتم إنشاؤه من خلال البرنامج التالي :

```
*****English.prg
* sets up memory file for storing English equivalents.
CLEAR
? "Creating English.mem file with English for numbers"
?
SET DEFAULT TO C
SET TALK ON
CLEAR MEMORY

U = " "
U1 = "ONE"
U2 = "TWO"
U3 = "TREE "
U4 = "FOUR "
U5 = "FIVE "
U6 = "SIX "
U7 = "SEVEN "
U8 = "EIGHT "
U9 = "NINE "
U10 = "TEN "
U11 = "ELEVEN "
U12 = "TWELVE "
U13 = "THIRTEEN "
U14 = "FOURTEEN "
U15 = "FIFTEEN "
U16 = "SIXTEEN "
U17 = "SEVENTEEN"
U18 = "EIGHTEEN"
U19 = "NINETEEN"
U20 = "TWENTY"
```

```
U30 = "THIRTY"
U40 = "FOURTY"
U50 = "FIFTY"
U60 = "SIXTY"
U70 = "SEVENTY"
U80 = "EIGHTY"
U90 = "NINETY"
```

```
* - - - - - Save all variables to English.mem file .
SAVE TO English
CLEAR
?
?
?
SET TALK OFF
RETURN
```

وهذا البرنامج يؤدي إلى إنشاء ملف الذاكرة (English.mem) الذي يجب تحميله في الذاكرة عند تشغيل برنامج إنشاء الشيكات . ويلاحظ أن كل متغير يبدأ اسمه بالحرف (U) يليه رقم يمثل الرقم المطلوب تحويله فمثلا (U5) يمثل (Five) ، (U30) يمثل (Thirty) . وهكذا .

ولتحميل هذا الملف في الذاكرة يتم كتابة السطر التالي :

```
RESTORE FROM English
```

والخطوة التالية بعد ذلك هي إنشاء ملف الخطوات (Procedure File) أو البرنامج الذي يقوم بترجمة أي عدد إلى المقابل الحرفي له . وهذا البرنامج نسميه (Translat.prg) ويتم كتابته كالآتي :

```
*****•Translat.prg
* - - - - - Procedure to convert a number to English
* - - - - - equivalent.
PROCEDURE Translat
PARAMETERS Number, English
```

```

*      Set up memory variables
Counter = 1
Start = 1
String = STR(Amount, 9, 2)
* - - - - - Loop through thousands and hundreds
DO WHILE Counter < 3
    * - - - Split out hundreds , tens and ones.
    Chunk = SUBSTR(String, Start , 3)
    Hun = SUBSTR(Chunk , 1, 1)
    Ten = SUBSTR (Chunk, 2, 2)
    One = SUBSTR(Chunk , 3, 1)

    * - - - - - Handle hundreds portion.
    IF VAL(Chunk) > 99
        English = English + U & Hun + "Hundreds"
    ENDIF
    * - - - - - Handle second 2 digits.
    T = VAL(Ten)
    IF T > 0
        DO CASE
            * - - - Case 1 : handle even tens and teens;
            CASE (INT(T/10.0) = T/10.0) .OR. (T > 9 ;
                AND. T < 20)
                English = English + U & Ten
            * - - Case 2 : Handle greater than 10 but
            * - - - not evenly divisable.
            CASE T > 9 .AND. (INT(T/10.0) # T/10.0)
                Ten = SUBSTR(Ten, 1,1) + '0'
                English = English + U & Ten + " " + U&one
            * - - - Case 3 : Handle less than 10
            CASE T < 10
                English = English + U & One
        ENDCASE
    ENDIF (T > 0)
    * - - - - - Add "Thousand" if necessary

```

```

IF Amount > 999.99 .AND. Counter = 1
    English = English + "THOUSAND"
ENDIF (need to add "Thousand")
* - - - - - Prepare for pass through hundreds.
Start = 1
Counter = Counter + 1
ENDDO (while counter < 3)
* - - - - - Take out ratios
IF INT(Amount) > 0
    English = English + "And"
ENDIF
English = English + SUBSTR(String, 8 , 2) + "/100"

```

ويمكن استخدام هذا البرنامج داخل أى برنامج آخر عن طريق كتابة السطر التالى :

SET PROCEDURE TO Translate

ويؤدى هذا الى فتح ملف الخطوات السابق . ويجب قبل تشغيل البرنامج تعريف المعاملات التى سوف تستخدم معه . وذلك كالآتى :

```

M_Amount = 9845
English = " "
DO Translat WITH M_Amount , English

```

ويجب قبل ذلك فتح ملف الذاكرة (English) وذلك بكتابة السطر التالى :

RESTORE FROM English

وبذلك يمكن اختبار البرنامج . ولكتابة البرنامج الذى يؤدى الى طباعة الشيك بالكامل يتم كتابة السطور التالية :

```

*****Checks.prg
* - - - - - Sample program to write checks

```

```

CLEAR
SET TALK OFF
SET SAFETY OFF
SET PROCEDURE TO Translat
* - - - - - Bring English equivalent variables.
RESTORE FROM English
* - - - - - Use the checks database
USE Checks
INDEX ON Check_No TO CheckNo
?
? "First Check to be printed will be", Check_No
?
?
WAIT "Press any key to begin writing checks"
CLEAR
SET PRINT ON
GO TOP
DO WHILE .NOT. EOF()
    * - - - - - Translate Amount to English
    M_Amount = Amount
    English = " "

    DO Translat WITH M_Amount , English
    * - - - Print the check with required modification
    # - - - - - for the check
    ? " " " Date
    ?
    ? To_Whom, " " , Amount
    ?
    ? English
    ?
    ?
    ?
    SKIP
ENDDO (not eof)

```

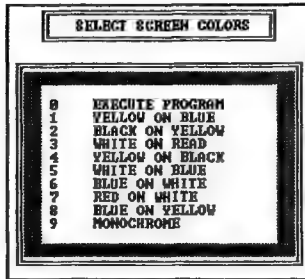
SET PRINT OFF
CLOSE PROCEDURE
CLEAR MEMORY
CLOSE DATABASES
RETURN

هذا البرنامج يزدى الى طباعة شيكات تحتوى على المبلغ (Amount) مكتوبا
بالارقام وكذلك بالحروف .

الفصل الخامس والعشرون

برنامج اختيار الألوان

هذا البرنامج يسمح للمستخدم باختيار ألوان الشاشة وكذلك ألوان الأعمدة الضوئية (Highlights). وذلك عن طريق عرض قائمة بالألوان المختلفة والسماح للمستخدم باختيار الألوان المطلوبة أنظر الشكل (٢٥ - ١).



شكل (٢٥ - ١)

ويمكن تسمية هذا البرنامج (Colors.prg) ويتم كتابته كالآتي :

```
*****Colors.prg
* - - - - - to give the user the colors he needs.
CLEAR
SET TALK OFF
SET STATUS OFF
* - - - - - Set up loop for displaying the menu
DO WHILE .T.
    * - - - - - Draw a box
    @ 1,24 TO 3,50 DOUBLE
    * - Set up loop for drawing a frame inside the box
    R = 6
```

```

DO WHILE R > 5 .AND. R < 20
    @ R,22 SAY REPLICATE(CHR(178),31)
    R = R + 1
ENDDO

* - - - - Erase an area from the box
@ 7, 24 SAY CLEAR TO 18,50
@ 5, 21 TO 20,53 DOUBLE
* - - - - Display the menu."
@ 2, 28 SAY "Select Screen Colors"
@ 8, 25 SAY " 0 - Execute Program"
@ 9, 25 SAY " 1 - Yellow On Blue"
@ 10,25 SAY " 2 - Black On Yellow"
@ 11,25 SAY " 3 - White On Red"
@ 12,25 SAY " 4 - Yellow On Black"
@ 13,25 SAY " 5 - White On Blue"
@ 14,25 SAY " 6 - Blue On White"
@ 15,25 SAY " 7 - Red On White"
@ 16,25 SAY " 8 - Blue On Yellow"
@ 17,25 SAY " 9 - Monochrom"
WAIT " " TO P

DO CASE
    CASE P = 0
        RETURN
    CASE P = 1
        COLSTR = "GR + /B, W/R, GR"
    CASE P = 2
        COLSTR = "N/GR, W/R, GR"
    CASE P = 3
        COLSTR = "W/R , W/N"
    CASE P = 4
        COLSTR = " GR + /N , W/R , GR"
    CASE P = 5
        COLSTR = " W/B , W/R , B"
    CASE P = 6
        COLSTR = " B/W , W/R , R"
    CASE P = 7
        COLSTR = "R/W , W/N , B"

```

```

CASE P = 8
    COLSTR = "B/GR , W/R , B"
CASE P = 9
    COLSTR = "7/0 , 0/7 , 0"
ENDCASE

SET COLOR TO & COLSTR
CLEAR
ENDDO

```

ويلاحظ في بداية البرنامج استخدام حلقة تكرارية لرسم مستطيل داخلي وملؤه بالتظليل باستخدام الحرف (CHR(178)) ، ثم استخدام الأمر (CLEAR) بعد ذلك لمسح الجزء الداخلي من هذا المستطيل حتى يظهر كإطار حول قائمة الاختيارات .

كما يلاحظ أن الألوان التي تظهر على الشاشة (COLSTR) تعتمد على الرقم المخزن في متغير الذاكرة (P) والذي تم اختياره من قائمة الألوان . وهذا البرنامج يمكن استخدامه مع أي برنامج آخر بكتابة السطر التالي :

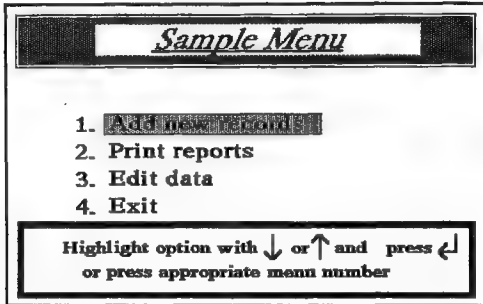
```
DO Colors
```


الفصل السادس والعشرون

برنامج تحريك العمود الضوئي

هذا البرنامج يستخدم عندما يريد مخطط البرامج اضافة مزيد من الاثارة والتشويق الى قوائم الاختيارات . حيث يمكن عند تنفيذه عرض عمود ضوئي (Highlight) يمكن تحريكه بواسطة مفاتيح الاتجاهات لأعلى ولأسفل () للوصول الى الاختيار المطلوب ثم الضغط على مفتاح الادخال لتنفيذ هذا الاختيار . وفي نفس الوقت يتيح للمستخدم الاختيار عن طريق كتابة الرقم الموجود عند أي اختيار .

وعند تنفيذ هذا البرنامج تظهر الشاشة التالية :



شكل (٢٦ - ١)

والسطور التالية توضح نموذجاً لبرنامج (Sample.prg) يستخدم هذه الطريقة في عرض القائمة الرئيسية :

```
*****Sample.prg
*   Create a sample menu with moving "light bar".
CLEAR
SET TALK OFF
```

```

* - - - - - create menu options (opt1-opt4)
opt1 = "1. Add new records"
opt2 = "2. Print reports"
opt3 = "3. Edit data"
opt4 = "4. Exit"

* - - - - - Display the menu
@ 1,1 TO 3,79 DOUBLE
@ 2,32 SAY "Sample menu"
@ 5,30 SAY Opt1
@ 6,30 SAY Opt2
@ 7,30 SAY Opt3
@ 8,30 SAY Opt4

* - - Display instructions with graphics characters.
@ 14,1 TO 18,78 DOUBLE
@ 15,18 SAY "Highlight option by using"
@ 15,40 SAY CHR(24) + "or" + CHR(25) + "and press"
      + CHR(17) + CHR(217)
@ 17,22 SAY "or press appropriate menu number"

* - - - - - Initialize memory variables.
Opt = 1
Sub = STR(Opt,1)
KeyPress = 0
Choice = 0
* - - - - - Reverse video on option 1
@ 5,30 GET Opt1
CLEAR GETS

* - - - - - Loop for choosing menu options.
DO WHILE Choice # 4
  * - - - - - Wait for a keyPress.
  KeyPress = 0
  DO WHILE KeyPress = 0
    KeyPress = INKEY()
  ENDDO (KeyPress)

```



```

* - - - - - Arrow key pressed.
IF KeyPress = 24 .OR. KeyPress = 5
    @ Opt+4,30 SAY Opt&Sub
    Opt = IIF(KeyPress = 24, Opt+1, Opt - 1)
    Opt = IIF(Opt > 4 , 1, Opt)
    Opt = IIF(Opt < 1 , 4, Opt)
    Sub = STR(Opt,1)
    @ Opt+4,30 GET Opt &Sub
    CLEAR GETS
    LOOP
ENDIF

* - - - - - Option numbers entered.
IF KeyPress >= 49 .AND. KeyPress <= 52
    Choice = KeyPress - 48
ENDIF

* - - - - - Return pressed
IF KeyPress = 13
    Choice = opt
ENDIF

* - - - - - An option was selected.
IF Choice > 0
    CLEAR
    ? 'Choice' , Choice
    Choice = 5
ENDIF
ENDDO(Choice)

```

والبرنامج يبدأ بتخزين كل سطر من سطور القائمة في متغير ذاكرة وذلك يتضح من السطور التالية :

```
* - - - - - create menu options (opt1-opt4)
opt1 = "1. Add new records"
Opt2 = "2. Print reports"
Opt3 = "3. Edit data"
Opt4 = "4. Exit"
```

والجزء التالي من البرنامج يؤدي الى ظهور قائمة الاختيارات على الشاشة ويتكون من السطور التالية :

```
* - - - - - Display the menu
@ 1,1 TO 3,79 DOUBLE
@ 2,32 SAY "Sample menu"
@ 5,30 SAY Opt1
@ 6,30 SAY Opt2
@ 7,30 SAY Opt3
@ 8,30 SAY Opt4
```

والجزء التالي من البرنامج يتم عن طريقه عرض لوحة المساعدة (Help) التي تظهر على الشاشة لتوضح للمستخدم كيفية الاختيار من القائمة . ويلاحظ استخدام حروف الرسم (Graphic Characters) التي يتم الحصول عليها باستخدام الدالة (CHR) . حيث يتم كتابة السهم العلوى () باستخدام الدالة CHR(24) ويتم كتابة السهم لأسفل باستخدام الدالة CHR(25) .

ويتم استخدام الدالة CHR(17) في كتابة حروف رسم معينة تمثل شكل مفتاح الادخال () ، وهكذا . ويتضح ذلك من السطور التالية :

```
@ 14,1 TO 18,78 DOUBLE
@ 15,18 SAY "Highlight option by using"
@ 15,40 SAY CHR(24) + "or" + CHR(25) + "and press" ;
+ CHR(17) + CHR(217)
@ 17,22 SAY "or press appropriate menu number"
```

وفي الجزء التالي يتم انشاء مجموعة من متغيرات الذاكرة مثل المتغير (Opt) الذى يتم فيه تخزين الرقم الذى يتم اختياره ، والمتغير (Sub) الذى يتم فيه تخزين القيمة الحرفية لهذا الرقم ، والمتغير (KeyPress) الذى يتم فيه تخزين كود الأسكى الخاص بالمفتاح الذى يضغط عليه المستخدم ، والمتغير (Choice) الذى يتم فيه تخزين الاختيار المطلوب لاستخدامه بعد ذلك فى التفرع الى البرنامج

الخاص به . ويتم ذلك من خلال السطور التالية :

```
Opt = 1
Sub = STR(Opt,1)
KeyPress = 0
Choice = 0
```

والجزء التالي من البرنامج يؤدي الى ظهور عمود ضوئي في المكان الممثل للاختيار رقم (1) في القائمة وذلك باستخدام الأمر (GET) . كما يتم مسح المتغير (GET) الذي يتكون نتيجة لذلك مع بقاء العمود الضوئي على هذا الاختيار . ويتم ذلك من خلال السطرين التاليين :

```
@ 5,30 GET Opt1
CLEAR GETS
```

والجزء التالي يتم من خلاله تكوين حلقة تكرارية لعرض القائمة دائما على الشاشة حتى يختار المستخدم الخروج . وذلك من خلال السطر التالي :

```
DO WHILE Choice # 4
```

كما تستخدم حلقة تكرارية اخرى لانتظار ضغط المستخدم على اى مفتاح . وفي هذه الحالة يتم تخزين كود الأسكى (ASCII Code) الخاص بهذا المفتاح في متغير الذاكرة (KeyPress) . وتستخدم الدالة INKEY() في الحصول على كود الأسكى الخاص بآخر مفتاح قام المستخدم بالضغط عليه . ويتضح ذلك من السطور التالية :

```
KeyPress = 0
DO WHILE KeyPress = 0
    KeyPress = INKEY()
ENDDO (KeyPress)
```

وعند ضغط المستخدم على مفتاح السهم لأعلى ((CHR(24)) أو مفتاح السهم لأسفل ((CHR(25)) فإن ذلك يؤدي الى اختفاء العمود الضوئي من الاختيار رقم (١) . وذلك لأن الأمر (SAY) يؤدي الى اعادة كتابة السطر فوق العمود الضوئي . كما يؤدي الى اضافة واحد أو طرح واحد من الرقم المخزن في المتغير (Opt) . ونلاحظ هنا استخدام الدالة (IIF) في زيادة قيمة المتغير (Opt) أو انقاصه حسب القيمة المخزنة في المتغير (KeyPress) نتيجة ضغط المستخدم على مفتاح معين . ويتضح ذلك من السطور التالية :

```
IF KeyPress = 24 .OR. KeyPress = 25
  @ Opt+4,30 SAY Opt&Sub
  Opt = IIF(KeyPress = 24, Opt+1, Opt - 1)
  Opt = IIF(Opt > 4 , 1, Opt)
  Opt = IIF(Opt < 1 , 4, Opt)
  Sub = STR(Opt,1)
  @ Opt+4, 30 GET Opt&Sub
  CLEAR GETS
  LOOP
ENDIF
```

وعندما يريد المستخدم الاختيار بالرقم وليس عن طريق تحريك العمود الضوئي فانه يكتب رقما بين (1) ، (4) . وحيث أن كود الأسكى الخاص بالأرقام يبدأ من (48) لذلك فان قيمة العدد الذي يختاره المستخدم يمكن حسابها بطرح العدد (48) من العدد الممثل لكود الأسكى (ASCII) الخاص بهذا العدد . ويتم ذلك من خلال السطور التالية :

```
* - - - - Option numbers entered.
IF KeyPress > = 49 .AND. KeyPress < = 52
  Choice = KeyPress - 48
ENDIF
```

وعندما يضغط المستخدم على مفتاح الادخال ((CHR(13)) يتم تخزين العدد الموجود في المتغير (Opt) في المتغير (Choice) . ويتم ذلك من خلال السطور التالية :

```
IF KeyPress = 13
  Choice = opt
ENDIF
```

وسواء كتب المستخدم رقم الاختيار المطلوب أو استخدم مفاتيح الاتجاهات في تحريك العمود الضوئي ثم ضغط على مفتاح الادخال فان المتغير (Choice) يتم فيه تخزين رقم معين يمثل هذا الاختيار .

وفي برنامج القائمة العادية يتم استخدام الأمر (DO CASE) في التفرع الى برنامج

معين بناء على الرقم الموجود في المتغير (Choice) . أما في هذا البرنامج فقد
تم الاكتفاء بمرض الرقم الموجود في المتغير (Choice) حتى يتم اختبار البرنامج
والتأكد من تحقيقه للمطلوب .

موسوعة "دلتا كمبيوتر" لتكنولوجيا وعلوم الحاسب

تعتمد المكتبة العربية ومحتوياتها في مجال التكنولوجيا من أكبر الدعائم الأساسية للمعرفة والتي تشكل بدورها أحد العوامل الرئيسية لجوانب التنمية المختلفة في المنطقة العربية . ولما كانت تكنولوجيا الحاسبات من أهم المجالات المعرفة التكنولوجية في الآونة الأخيرة فإن قيمة المؤلفات تزداد في هذا الجانب من واقع ازدياد حاجة المستخدم العربي إليها . ولما لاشك فيه أن المكتبة العربية في مجال تكنولوجيا وعلوم الحاسب تعتبر ثقبرة في هذا النوع من المؤلفات إلى درجة بحيث نقرأ لعدة جوانب نذكر منها مايلي :

- العنق الفنى اللازم والمراكيب التطوير التكنولوجى السريع .
 - انفتاح المكتبة العربية الى القدر المطلوب من البعد العلمى اللازم للبعد الفنى .
 - الترابط الكامل بين جوانب المعرفة في المراجع المختلفة وعلقات ذلك بدرجة استفادة القارئ وانعكاسه على درجة المعرفة ومستوى الخبرة .
 - درجة ارتباطها بالتطبيق ومستوى استفادة القارئ منها .
 - النقطية الكاملة لكل مستويات القراء مع اختلاف ثقافتهم وخبراتهم .
 - حاجة القارئ العربى فى هذه المرحلة -تجاوز مستوى العديد من المراجع المتاحة والتي تعتمد على الترجمة المرفقة لدليل التشغيل للنظم التكنولوجية المختلفة الخاصة بالحاسب .
- ومن هذا المنطلق فقد قامت مؤسسة دلتا بإعداد موسوعة "دلتا كمبيوتر" لتكنولوجيا وعلوم الحاسب - والتي تتكون من العديد من المراجع - على أبنى نغمة مختارة من أساطلة الجامعات وكبار الخبراء المتخصصين فى هذا المجال .
- ومع التطور السريع فى عالم تكنولوجيا الحاسبات وتعدد جوانب المعرفة المطلوبة للقارئ العربى فإن موسوعة دلتا قد تم إعدادها على أساس النقطية الشاملة لاجهات التكنولوجيا الحديثة تبعاً للأولويات المطروحة مع النقطية المستمرة للمستجدات فى هذا المجال من خلال الاصدارات المختلفة لكتب الموسوعة على ضوء التطور السريع فى مجال تكنولوجيا الحاسبات .

١ - الحاسبات الالكترونية حاضرها ومستقبلها

٢ - الموسوعة الشاملة لمصطلحات الحاسب الالكترونى

٣ - المرجع الشامل لنظام التشغيل (DOS)
MS DOS
MS WINDOWS NORTON UTILITIES
PC TOOLS VIRUS-SCAN

٤ - عالم الجداول الالكترونية
LOTUS 123
EXCEL
QUATRO PRO

٥ - نظم إدارة قواعد البيانات (الجزء الاول)
FOXBASE+ DBASE III+
FOXPRO DBASE IV

٦ - نظم إدارة قواعد البيانات (الجزء الثانى)

٧ - تطبيقات نظم إدارة قواعد البيانات

٨ - فيروسات الحاسب وأمن البيانات

٩ - النظم المحاسبية والحاسب الالكترونى
حسابات الميزان - حسابات العملة
حسابات الميزان - حسابات الميزان
حسابات الميزان

١٠ - الحاسب الالكترونى وإدارة المشروعات

١١ - النظم الخبيرة والدكاء الاصطناعى

دلتا كمبيوتر
Delta Computer

تطبيقات نظم إدارة قواعد البيانات

- ١٣ - برنامج القائمة الرئيسية
- ١٤ - برنامج تشغيل الملف الرئيسي
- ١٥ - برنامج تشغيل ملف المبيعات
- ١٦ - برنامج تشغيل ملف الاضافة
- ١٧ - برنامج تحديث البيانات

الجزء الرابع

نظام حسابات العملاء

- ١٨ - تصميم النظام
- ١٩ - ملفات الخطوات
- ٢٠ - برنامج القائمة الرئيسية والادخال والتعديل

- ٢١ - تقارير برنامج حسابات العملاء
- ٢٢ - التحديث الشهري للنظام
- ٢٣ - برنامج التكامل بين حسابات العملاء والمخازن

الجزء الخامس

بعض الأدوات المتقدمة

- ٢٤ - برنامج كتابة الشيكات
- ٢٥ - برنامج اختيار الألبان
- ٢٦ - برنامج تحريك العمود الضوئي

- ١ - مقدمة

الجزء الأول

نظم إدارة قواعد البيانات

- ٢ - انواع البرامج
- ٣ - البحث السريع
- ٤ - خطوات تصميم النظام
- ٥ - كتابة البرامج
- ٦ - وسائل التصحيح

الجزء الثاني

نظام معلومات الطلبة

- ٧ - تصميم النظام
- ٨ - البرنامج الرئيسي
- ٩ - برنامج التقارير
- ١٠ - برنامج التصحيح
- ١١ - برنامج مسح السجلات

الجزء الثالث

نظام المخازن

- ١٢ - توصيف النظام